Reference Manual

Allen-Bradley

Logix5000 Controllers Motion Instructions

Catalog Numbers 1756 ControlLogix, 1756 GuardLogix, 1768 CompactLogix, 1789 SoftLogix

Allen-Bradley • Rockwell Software

Rockwell Automation

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

	WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
	ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.

	SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
	BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.
	ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Allen-Bradley, CompactLogix, ControlLogix, DriveLogix, GuardLogix, Kinetix, Logix5000, PowerFlex, Rockwell Automation, Rockwell Software, RSLogix 5000, and SoftLogix, are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

This manual contains new and updated information. Changes throughout this revision are marked by change bars, as shown to the right of this paragraph.

Updated Information

The document contains these changes.

Topic	Page
Updated Additional Resources table	33
Updated text and formatting	Throughout
Added AXIS_CIP_DRIVE type to Table 44	123
Updated Error 75 Corrective Action or Cause	349
Updated table footnote	393

Notes:

Motion Instruction Locator

Where to Find an Instruction

Use this locator to find the reference details about Logix instructions. The grayed-out instructions are available in other manuals. This locator also lists which programming languages are available for the instructions.

Table 1 - Instruction Locator

If the locator lists	The instruction is documented in
A page number	This manual
Coordinate	Motion Coordinate System User Manual, MOTION-UM002
General	Logix5000™ Controllers General Instructions Reference Manual, 1756-RM003
Process	Logix5000 Controllers Process Control and Drives Instructions Reference Manual, 1756-RM006

Instruction	Location	Languages
ABL ASCII Test For Buffer Line	general	relay ladder structured text
ABS Absolute Value	general	relay ladder structured text function block
ACB ASCII Chars in Buffer	general	relay ladder structured text
ACL ASCII Clear Buffer	general	relay ladder structured text
ACOS Arc Cosine	general	structured text
ACS Arc Cosine	general	relay ladder function block
ADD Add	general	relay ladder structured text function block
AFI Always False Instruction	general	relay ladder
AHL ASCII Handshake Lines	general	relay ladder structured text
ALM Alarm	process	structured text function block
ALMA Analog Alarm	general	relay ladder structured text function block
ALMD Digital Alarm	general	relay ladder structured text function block
AND Bitwise AND	general	relay ladder structured text function block
ARD ASCII Read	general	relay ladder structured text

Instruction	Location	Languages
ARL ASCII Read Line	general	relay ladder structured text
ASIN Arc Sine	general	structured text
ASN Arc Sine	general	relay ladder function block
ATAN Arc Tangent	general	structured text
ATN Arc Tangent	general	relay ladder function block
AVE File Average	general	relay ladder
AWA ASCII Write Append	general	relay ladder structured text
AWT ASCII Write	general	relay ladder structured text
BAND Boolean AND	general	structured text function block
BNOT Boolean NOT	general	structured text function block
BOR Boolean OR	general	structured text function block
BPT Breakpoints	general	relay ladder
BRK Break	general	relay ladder
BSL Bit Shift Left	general	relay ladder
BSR Bit Shift Right	general	relay ladder
BTD Bit Field Distribute	general	relay ladder

Instruction	Location	Languages
BTDT Bit Field Distribute with Target	general	structured text function block
BTR Message	general	relay ladder structured text
BTW Message	general	relay ladder structured text
BXOR Boolean Exclusive OR	general	structured text function block
CC Coordinated Control	process	structured text function block
CLR Clear	general	relay ladder structured text
CMP Compare	general	relay ladder
CONCAT String Concatenate	general	relay ladder structured text
COP Copy File	general	relay ladder structured text
COS Cosine	general	relay ladder structured text function block
CPS Synchronous Copy File	general	relay ladder structured text
CPT Compute	general	relay ladder
CTD Count Down	general	relay ladder
CTU Count Up	general	relay ladder
CTUD Count Up/Down	general	structured text function block
D2SD Discrete 2-State Device	process	structured text function block
D3SD Discrete 3-State Device	process	structured text function block
DDT Diagnostic Detect	general	relay ladder
DEDT Deadtime	process	structured text function block
DEG Degrees	general	relay ladder structured text function block
DELETE String Delete	general	relay ladder structured text
DERV Derivative	process	structured text function block

Instruction	Location	Languages
DFF D Flip-Flop	process	structured text function block
DIV Divide	general	relay ladder structured text function block
DTOS DINT to String	general	relay ladder structured text
DTR Data Transitional	general	relay ladder
EOT End of Transition	general	relay ladder structured text
EQU Equal to	general	relay ladder structured text function block
ESEL Enhanced Select	process	structured text function block
EVENT Trigger Event Task	general	relay ladder structured text
FAL File Arithmetic and Logic	general	relay ladder
FBC File Bit Comparison	general	relay ladder
FFL FIFO Load	general	relay ladder
FFU FIFO Unload	general	relay ladder
FGEN Function Generator	process	structured text function block
FIND Find String	general	relay ladder structured text
FLL File Fill	general	relay ladder
FOR For	general	relay ladder
FRD Convert to Integer	general	relay ladder function block
FSC File Search and Compare	general	relay ladder
GEQ Greater than or Equal to	general	relay ladder structured text function block
GRT Greater Than	general	relay ladder structured text function block
GSV Get System Value	general	relay ladder structured text

Instruction	Location	Languages
HLL High/Low Limit	process	structured text function block
HPF High Pass Filter	process	structured text function block
ICON Input Wire Connector	process	function block
IMC Internal Model Control	process	structured text function block
INSERT Insert String	general	relay ladder structured text
INTG Integrator	process	structured text function block
IOT Immediate Output	general	relay ladder structured text
IREF Input Reference	process	function block
JKFF JK Flip-Flop	process	structured text function block
JMP Jump to Label	general	relay ladder
JSR Jump to Subroutine	general	relay ladder structured text function block
JXR Jump to External Routine	general	relay ladder
LBL Label	general	relay ladder
LDL2 Second-Order Lead Lag	process	structured text function block
LDLG Lead-Lag	process	structured text function block
LEQ Less Than or Equal to	general	relay ladder structured text function block
LES Less Than	general	relay ladder structured text function block
LFL LIFO Load	general	relay ladder
LFU LIFO Unload	general	relay ladder
LIM Limit	general	relay ladder function block
LN Natural Log	general	relay ladder structured text function block

Instruction	Location	Languages
LOG Log Base 10	general	relay ladder structured text function block
LOWER Lower Case	general	relay ladder structured text
LPF Low Pass Filter	process	structured text function block
MAAT Motion Apply Axis Tuning	261	relay ladder structured text
MAFR Motion Axis Fault Reset	75	relay ladder structured text
MAG Motion Axis Gear	123	relay ladder structured text
MAH Motion Axis Home	92	relay ladder structured text
MAHD Motion Apply Hookup Diagnostics	278	relay ladder structured text
MAJ Motion Axis Jog	99	relay ladder structured text
MAM Motion Axis Move	111	relay ladder structured text
MAOC Motion Arm Output Cam	239	relay ladder structured text
MAPC Motion Axis Position Cam	156	relay ladder structured text
MAR Motion Arm Registration	227	relay ladder structured text
MASD Motion Axis Shutdown	48	relay ladder structured text
MAS Motion Axis Stop	82	relay ladder structured text
MASR Motion Axis Shutdown Reset	54	relay ladder structured text
MATC Motion Axis Time Cam	178	relay ladder structured text
MAVE Moving Average	process	structured text function block
MAW Motion Arm Watch	216	relay ladder structured text
MAXC Maximum Capture	process	structured text function block
MCD Motion Change Dynamics	133	relay ladder structured text
MCCD Motion Coordinated Change Dynamics	coordinate	relay ladder structured text
MCCM Motion Coordinated Circular Move	coordinate	relay ladder structured text

Instruction	Location	Languages
MCLM Motion Coordinated Linear Move	coordinate	relay ladder structured text
MCCP Motion Calculate Cam Profile	150	relay ladder structured text
MCR Master Control Reset	general	relay ladder
MCS Motion Coordinated Stop	coordinate	relay ladder structured text
MCSD Motion Coordinated Shutdown	coordinate	relay ladder structured text
MCSR Motion Coordinated Shutdown Reset	coordinate	relay ladder structured text
MCSV Motion Calculate Slave Values	192	relay ladder structured text
MCT Motion Coordinated Transform	coordinate	relay ladder structured text
MCTP Motion Calculate Transform Position	coordinate	relay ladder structured text
MDAC	297	relay ladder structured text
MDCC Master Driven Coordinated Control	coordinate	relay ladder structured text
MDF Motion Direct Drive Off	65	relay ladder structured text
MDOC Motion Disarm Output Cam	256	relay ladder structured text
MDO Motion Direct Drive On	59	relay ladder structured text
MDR Motion Disarm Registration	236	relay ladder structured text
MDS Motion Drive Start	69	relay ladder structured text
MDW Motion Disarm Watch	223	relay ladder structured text
MEQ Mask Equal to	general	relay ladder structured text function block
MGSD Motion Group Shutdown	202	relay ladder structured text
MGS Motion Group Stop	196	relay ladder structured text
MGSP Motion Group Strobe Position	210	relay ladder structured text
MGSR Motion Group Shutdown Reset	206	relay ladder structured text
MID Middle String	general	relay ladder structured text

Instruction	Location	Languages
MINC Minimum Capture	process	structured text function block
MMC Modular Multivariable Control	process	structured text function block
MOD Modulo	general	relay ladder structured text function block
MOV Move	general	relay ladder
MRAT Motion Run Axis Tuning	268	relay ladder structured text
MRHD Motion Run Hookup Diagnostics	284	relay ladder structured text
MRP Motion Redefine Position	142	relay ladder structured text
MSF Motion Servo Off	43	relay ladder structured text
MSG Message	general	relay ladder structured text
MSO Motion Servo On	39	relay ladder structured text
MSTD Moving Standard Deviation	process	structured text function block
MUL Multiply	general	relay ladder structured text function block
MUX Multiplexer	process	function block
MVM Masked Move	general	relay ladder
MVMT Masked Move with Target	general	structured text function block
NEG Negate	general	relay ladder structured text function block
NEQ Not Equal to	general	relay ladder structured text function block
NOP No Operation	general	relay ladder
NOT Bitwise NOT	general	relay ladder structured text function block
NTCH Notch Filter	process	structured text function block
OCON Output Wire Connector	process	function block

Instruction	Location	Languages
ONS One Shot	general	relay ladder
OR Bitwise OR	general	relay ladder structured text function block
OREF Output Reference	process	function block
OSF One Shot Falling	general	relay ladder
OSFI One Shot Falling with Input	general	structured text function block
OSR One Shot Rising	general	relay ladder
OSRI One Shot Rising with Input	general	structured text function block
OTE Output Energize	general	relay ladder
OTL Output Latch	general	relay ladder
OTU Output Unlatch	general	relay ladder
PATT Attach to Equipment Phase	phase	relay ladder structured text
PCLF Equipment Phase Clear Failure	phase	relay ladder structured text
PCMD Equipment Phase Command	phase	relay ladder structured text
PDET Detach from Equipment Phase	phase	relay ladder structured text
PFL Equipment Phase Failure	phase	relay ladder structured text
PI Proportional + Integral	process	structured text function block
PID Proportional Integral Derivative	general	relay ladder structured text
PIDE Enhanced PID	process	structured text function block
PMUL Pulse Multiplier	process	structured text function block
POSP Position Proportional	process	structured text function block
POVR Equipment Phase Override Command	phase	relay ladder structured text
PPD Equipment Phase Paused	phase	relay ladder structured text
PRNP Equipment Phase New Parameters	phase	relay ladder structured text
PSC Phase State Complete	phase	relay ladder structured text
PXRQ Equipment Phase External Request	phase	relay ladder structured text
RAD Radians	general	relay ladder structured text function block
RES Reset	general	relay ladder
RESD Reset Dominant	process	structured text function block
RET Return	general	relay ladder structured text function block
RLIM Rate Limiter	process	structured text function block
RMPS Ramp/Soak	process	structured text function block
RTO Retentive Timer On	general	relay ladder
RTOR Retentive Timer On with Reset	general	structured text function block
RTOS REAL to String	general	relay ladder structured text
SBR Subroutine	general	relay ladder structured text function block
SCL Scale	process	structured text function block
SCRV S-curve	process	structured text function block
SEL Select	process	function block
SETD Set Dominant	process	structured text function block
SFP SFC Pause	general	relay ladder structured text
SFR SFC Reset	general	relay ladder structured text
SIN Sine	general	relay ladder structured text function block
SIZE Size In Elements	general	relay ladder structured text
SNEG Selected Negate	process	structured text function block

Instruction	Location	Languages
SOC Second-Order Controller	process	structured text function block
SQI Sequencer Input	general	relay ladder
SQL Sequencer Load	general	relay ladder
SQO Sequencer Output	general	relay ladder
SQR Square Root	general	relay ladder function block
SQRT Square Root	general	structured text
SRT File Sort	general	relay ladder structured text
SRTP Split Range Time Proportional	process	structured text function block
SSUM Selected Summer	process	structured text function block
SSV Set System Value	general	relay ladder structured text
STD File Standard Deviation	general	relay ladder
STOD String To DINT	general	relay ladder structured text
STOR String To REAL	general	relay ladder structured text
SUB Subtract	general	relay ladder structured text function block
SWPB Swap Byte	general	relay ladder structured text
TAN Tangent	general	relay ladder structured text function block
TND Temporary End	general	relay ladder
TOD Convert to BCD	general	relay ladder function block
TOF Timer Off Delay	general	relay ladder
TOFR Timer Off Delay with Reset	general	structured text function block
TON Timer On Delay	general	relay ladder
TONR Timer On Delay with Reset	general	structured text function block
TOT Totalizer	process	structured text function block
TPT Tracepoints	general	relay ladder
TRN Truncate	general	relay ladder function block
TRUNC Truncate	general	structured text
UID User Interrupt Disable	general	relay ladder structured text
UIE User Interrupt Enable	general	relay ladder structured text
UPDN Up/Down Accumulator	process	structured text function block
UPPER Upper Case	general	relay ladder structured text
XIC Examine If Closed	general	relay ladder
XIO Examine If Open	general	relay ladder
XOR Bitwise Exclusive OR	general	relay ladder structured text function block
XPY X to the Power of Y	general	relay ladder structured text function block

Preface

If You Have a PowerFlex 700S Drive with DriveLogix Controller 23
Who Should Use This Manual 23
Purpose of This Manual 24
Conventions and Related Terms 25
Set and Clear 25
Rung Condition 25
Sequential Function Chart (SFC) 26
Instruction Timing 26
Immediate Type Instructions 26
Message Type Instructions 28
Process Type Instructions 29
Choose a Command 31
Additional Resources 33

Chapter 1

Motion State Instructions (MSO, MSF, MASD, MASR, MDO, MDF, MDS, MAFR)

Motion Servo On (MSO) 39
Operands 39
Description 40
Arithmetic Status Flags 40
Fault Conditions 40
MSO Execution Conditions 41
Error Codes 41
MSO Changes to Status Bits 42
Example 42
Motion Servo Off (MSF) 43
Operands 43
Description 44
Arithmetic Status Flags 45
Fault Conditions 45
MSF Execution Conditions 46
Error Codes 46
MSF Changes to Status Bits 47
MSF Example 47
Motion Axis Shutdown (MASD) 48
Operands 48
Description 49
Arithmetic Status Flags 50
Fault Conditions 50
MASD Execution Conditions 51
Error Codes 52
MASD Changes to Status Bits 52
MASD Example 52
Motion Axis Shutdown Reset (MASR) 54
Operands 54

Description ... 55
Arithmetic Status Flags ... 56
Fault Conditions ... 56
MASR Execution Conditions ... 56
Error Codes ... 57
MASR Changes to Status Bits ... 57
MASR Example ... 58
Motion Direct Drive On (MDO) ... 59
Operands ... 59
Description ... 60
Loss of Feedback When Using an MDO Instruction ... 62
Arithmetic Status Flags ... 62
Fault Conditions ... 62
MDO Execution Conditions ... 63
Error Codes ... 64
MDO Changes to Status Bits ... 64
MDO Example ... 64
Motion Direct Drive Off (MDF) ... 65
Operands ... 65
Description ... 66
Arithmetic Status Flags ... 66
Fault Conditions ... 66
MDF Execution Conditions ... 67
Error Codes ... 68
MDF Changes to Status Bits ... 68
MDF Example ... 68
Motion Drive Start (MDS) ... 69
Operands ... 69
Description ... 70
Arithmetic Status Flags ... 73
Fault Conditions ... 74
Error Codes ... 74
MDS Changes to Status Bits ... 74
MDS Example ... 74
Motion Axis Fault Reset (MAFR) ... 75
Operands ... 75
Description ... 76
Arithmetic Status Flags ... 77
Fault Conditions ... 77
MAFR Execution Conditions ... 77
Error Codes ... 78
MAFR Changes to Status Bits ... 78
MAFR Example ... 78

Chapter 2

Motion Move Instructions (MAS, MAH, MAJ, MAM, MAG, MCD, MRP, MCCP, MAPC, MATC, MCSV)

Motion Axis Stop (MAS) 82
- Operands 82
- Description 85
- Programming Guidelines 86
- Arithmetic Status Flags 87
- Fault Conditions 87
- MAS Error Codes 87
- MAS Changes to Status Bits 88
- Example 1 89
- Example 2 90

Motion Axis Home (MAH) 92
- Operands 92
- Description 93
- Programming Guidelines 95
- Arithmetic Status Flags 95
- Fault Conditions 95
- MAH Execution Condition 96
- MAH Error Codes 97
- MAH Changes to Status Bits 98

Motion Axis Jog (MAJ) 99
- Operands 99
- Description 103
- Programming Guidelines 103
- Arithmetic Status Flags 105
- Fault Conditions 105
- MAJ Error Codes 105
- MAJ Changes to Status Bits 106
- Example 1 107
- Example 2 109

Motion Axis Move (MAM) 111
- Operands 111
- Description 114
- Programming Guidelines 115
- Arithmetic Status Flags 119
- Fault Conditions 119
- Error Codes 120
- MAM Changes to Motion Status Bits 120
- MAM Example 121

Motion Axis Gear (MAG) 123
- Operands 123
- Description 126
- MAG Programming Guidelines 127
- Arithmetic Status Flags 129
- Fault Conditions 129
- MAG Execution Conditions 130
- MAG Error Codes 131

MAG Changes to Status Bits 132
MAG Example 132
Motion Change Dynamics (MCD) 133
Operands 133
Description 136
Programming Guidelines 137
Arithmetic Status Flags 139
Fault Conditions 139
MCD Execution Conditions 139
MCD Error Codes 140
MCD Changes to Status Bits 140
MCD Example 141
Motion Redefine Position (MRP) 142
Operands 142
Description 143
Programming Guidelines 144
Arithmetic Status Flags 146
Fault Conditions 146
MRP Execution Conditions 147
MRP Error Codes 148
MRP Changes to Status Bits 148
MRP Example 149
Motion Calculate Cam Profile (MCCP) 150
Operands 150
Description 151
Specifying and Calculating the Cam Profile 152
Arithmetic Status Flags 154
Fault Conditions 154
MCCP Error Codes 154
MCCP Changes to Status Bits 154
MCCP Example 155
Motion Axis Position Cam (MAPC) 156
Operands 156
Description 160
Specifying the Cam Profile 161
Scaling Position Cams 162
MAPC Execution Schedule 163
Master Position Reference 169
Master Direction 170
Moving While Camming 171
Stopping a Cam 171
Merging from a Cam 172
Handling Axis Faults 172
Arithmetic Status Flags 173
Fault Conditions 173
MAPC Execution Conditions 174
MAPC Error Codes 175

MAPC Changes to Status Bits 176
MAPC Example 177
Motion Axis Time Cam (MATC) 178
Operands 178
Description 181
Specifying and Executing the Cam Profile 182
Scaling Time Cams 183
Execution Schedule 184
Stopping a Cam 187
Merging from a Cam 188
Arithmetic Status Flags 188
Fault Conditions 188
MATC Execution Conditions 189
MATC Error Codes 190
MATC Changes to Status Bits 191
MATC Example 191
Motion Calculate Slave Values (MCSV) 192
Operands 192
Description 193
Arithmetic Status Flags 193
MCSV Error Codes 194
MCSV Changes to Status Bits 194

Chapter 3

Motion Group Instructions (MGS, MGSD, MGSR, MGSP)

Motion Group Stop (MGS) 196
Operands 196
Description 197
Arithmetic Status Flags 199
Fault Conditions 199
MGS Execution Conditions 200
MGS Error Codes 201
MGS Changes to Status Bits 201
MGS Example 201
Motion Group Shutdown (MGSD) 202
Operands 202
Description 203
Arithmetic Status Flags 203
Fault Conditions 204
MGSD Execution Conditions 204
MGSD Error Codes 205
MGSD Changes to Status Bits 205
MGSD Example 205
Motion Group Shutdown Reset (MGSR) 206
Operands 206
Description 207
Arithmetic Status Flags 207
Fault Conditions 207

MGSR Execution Conditions 208
MGSR Error Codes 209
MGSR Changes to Status Bits 209
MGSR Example 209
Motion Group Strobe Position (MGSP) 210
Operands 210
Description 211
Arithmetic Status Flags 211
Fault Conditions 211
MGSP Execution Conditions 212
MGSP Error Codes 213
MGSP Changes to Status Bits 213
MGSP Example 213

Chapter 4

Motion Event Instructions (MAW, MDW, MAR, MDR, MAOC, MDOC)

Motion Arm Watch (MAW) 216
Operands 216
Description 218
Arithmetic Status Flags 219
Fault Conditions 219
MAW Execution Conditions 220
MAW Error Codes 222
. 222
MAW Changes to Status Bits 222
MAW Example 222
Motion Disarm Watch (MDW) 223
Operands 223
Description 224
Arithmetic Status Flags 224
Fault Conditions 224
MDW Execution Conditions 225
MDW Error Codes 226
MDW Changes to Status Bits 226
MDW Example 226
Motion Arm Registration (MAR) 227
Operands 227
Description 229
Rearming an MAR Instruction 230
Arithmetic Status Flags 231
Fault Conditions 232
MAR Execution Conditions 233
MAR Error Codes 234
MAR Changes to Status Bits 235
MAR Example 235
Motion Disarm Registration (MDR) 236
Operands 236
Description 237

Arithmetic Status Flags 237
Fault Conditions 237
Error Codes 237
MDR Changes to Status Bits 238
MDR Example 238
Motion Arm Output Cam (MAOC) 239
Operands 239
Description 242
Specifying the Output Cam 245
Specifying Output Compensation 248
Arithmetic Status Flags 251
Fault Conditions 251
MAOC Execution Conditions 252
Error Codes 252
MAOC Changes to Status Bits 254
Module Fault Conditions Disarm Output Cams 254
Axis and Module Fault Conditions Disarm Output Cams 254
MAOC Example 255
Motion Disarm Output Cam (MDOC) 256
Operands 256
Description 257
Arithmetic Status Flags 257
Fault Conditions 257
MDOC Execution Conditions 258
MDOC Error Codes 258
MDOC Changes to Status Bits 259
MDOC Example 260

Chapter 5

Motion Configuration Instructions (MAAT, MRAT, MAHD, MRHD)

Motion Apply Axis Tuning (MAAT) 261
Operands 262
Description 263
Arithmetic Status Flags 265
Fault Conditions 265
MAAT Execution Conditions 266
Error Codes 267
MAAT Changes to Status Bits 267
MAAT Example 267
Motion Run Axis Tuning (MRAT) 268
Operands 268
Description - AXIS_SERVO, AXIS_SERVO_DRIVE 269
Description - AXIS_CIP_DRIVE 272
Tune Status Parameter 274
Arithmetic Status Flags 275
Fault Conditions 275
MRAT Execution Conditions 276
Error Codes 277

MRAT Changes to Status Bits 277
MRAT Example 277
Motion Apply Hookup Diagnostics (MAHD) 278
Operands 278
Description 280
Motor Encoder Hookup Test 280
Encoder Hookup Test 281
Arithmetic Status Flags 281
Fault Conditions 281
MAHD Execution Conditions 282
MAHD Error Codes 283
MAHD Changes to Status Bits 283
Example 283
Motion Run Hookup Diagnostics (MRHD) 284
Operands 284
Description - AXIS_SERVO, AXIS_SERVO_DRIVE 286
Motor Encoder Hookup Test 287
Description - AXIS_CIP_DRIVE 289
Test Status 293
Arithmetic Status Flags 294
Fault Conditions 294
MRHD Execution Conditions 295
Error Codes 296
MRHD Changes to Status Bits 296
MRHD Example 296

Chapter 6

MDSC Functionality

Master Driven Axis Control (MDAC) 297
Operands 297
MOTION_INSTRUCTION 299
Interaction of MDAC with Time Cams (MATC) 300
Motion Control 301
Motion Direct Command and the MDAC Instruction 304
Arithmetic Status Flags 304
Fault Conditions 304
Error Codes 304
Master Driven and Time Driven Modes 305
Changing the Master Axis 306
Actions Taken When Stopping/Shutdown Instructions are Executed on the Slave Axis 307
Actions Taken When Stopping/Shutdown Instructions are Executed on the Master Axis 308
Master Driven Speed Control Parameters for Single Axis Motion Instructions 309
Acceleration and Deceleration Enumerations 318
Jerk Enumerations 319
Time Based Planning 323

Status Bits for Motion Instructions (MAM, MATC, MAJ) When MDAC is Active 326

Chapter 7

Tune an S-curve Profile

Procedure 328
Additional Resources 329

Chapter 8

Analyzing Axis Motion

Axis Accelerates While Stopping 331
Axis Overshoots the Target Speed 334
Delay If a Jog Is Stopped and Restarted 336
Axis Reverses Direction When Stopped and Started 338
Axis Overshoots Its Position and Reverses Direction 342

Appendix A

Error Codes (ERR) for Motion Instructions

Appendix B

Motion-related Data Types (Structures)

CAM Structure 353
CAM_PROFILE Structure 353
MOTION_GROUP Structure 354
MOTION_INSTRUCTION Data Type 356
OUTPUT_CAM Structure 357
OUTPUT_COMPENSATION Structure 358

Appendix C

Structured Text Programming

Structured Text Syntax 359
Assignments 360
Specify a Non-retentive Assignment 361
Assign an ASCII Character to a String 362
Expressions 362
Use Arithmetic Operators and Functions 363
Use Relational Operators 365
Use Logical Operators 366
Use Bitwise Operators 367
Determine the Order of Execution 368
Instructions 369
Constructs 370
Key Words Reserved for Future Use 370
IF...THEN 371
Operands 371
Description 371
Arithmetic Status Flags 372
Fault Conditions 372
Example 1 373

Example 2 373
Example 3 374
Example 4 374
CASE...OF 375
Operands 375
Description 376
Arithmetic Status Flags 376
Fault Conditions 376
Example 377
FOR...DO 378
Operands 378
Description 379
Arithmetic Status Flags 379
Fault Conditions 379
WHILE...DO 381
Operands 381
Description 381
Arithmetic Status Flags 382
Fault Conditions 382
REPEAT...UNTIL 384
Operands 384
Description 384
Arithmetic Status Flags 385
Fault Conditions 385
Comments 387

Appendix D

Motion Instructions and Integrated Motion Control Modes

Appendix E

Camming

Camming Concepts 393
Mechanical Camming 393
Electronic Camming 394
Cam Profiles 395
Position Cam Profile 395
Time Cam Profile 396
Calculating a Cam Profile 397
Using Common Cam Profiles 397
Acceleration Cam Profile 397
Run Cam Profile 398
Deceleration Cam Profile 399
Dwell Cam Profile 399
Dwell Cam Profile 400
Behavior of Pending Cams 400
Scaling Cams 401
Scaling Position Cam Profiles 401
Scaling Time Cam Profiles 402

Cam Execution Modes 403
Execution Schedule 403
MAPC Instruction 403
MATC Instruction 407
Pending Cams 408

Appendix F

MOTION-RM002D-EN-P 413
MOTION-RM002C-EN-P 413

History of Changes

Index

Notes:

This manual is one of several Logix5000-based instruction manuals.

Task	Resource
Configure Control Modes, Control Methods, and AXIS_CIP_DRIVE Attributes	Integrated Motion on the Ethernet/IP Network Reference Manual, publication MOTION-RM003
Configure and startup an integrated motion application	Integrated Motion on the Ethernet/IP Network Configuration and Startup User Manual, publication MOTION-UM003
Program the controller for motion applications	This manual, the Logix5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002
Import a text file or tags into a project	Logix5000 Controllers Import/Export Reference Manual, publication 1756-RM084
Export a project or tags to a text file	
Configure your controller for motion axes and motion modules	SERCOS and Analog Motion Configuration and Startup User Manual, publication MOTION-UM001
Configure your controller for coordinate system	Motion Coordinate System User Manual, publication MOTION-UM002

You can use these Logix5000 controllers for motion control:

- 1756 ControlLogix® controllers
- 1756 GuardLogix® controllers (version 16 and later)
- 1768 CompactLogix™ controllers (version 15 and later)
- 1769 CompactLogix controllers (version 20 and later)
- 1789 SoftLogix™5800 controllers
- 20D PowerFlex® 700S drive with DriveLogix™ controllers (version 19 and earlier)

If You Have a PowerFlex 700S Drive with DriveLogix Controller

You cannot use these instructions with a DriveLogix controller:

- Motion Direct Drive On (MDO)
- Motion Direct Drive Off (MDF)
- Motion Apply Axis Tuning (MAAT)
- Motion Run Axis Tuning (MRAT)
- Motion Apply Hookup Diagnostics (MAHD)
- Motion Run Hookup Diagnostics (MRHD)

Who Should Use This Manual

This document provides a programmer with details about the motion instructions that are available for a Logix5000 controller. You should already be familiar with how the Logix5000 controller stores and processes data. Novice programmers should read all the details about an instruction before using the instruction. Experienced programmers can refer to the instruction information to verify details.

Purpose of This Manual

This manual provides information about each motion instruction.

Topic	Provides this type of information
Instruction name	Identifies the instruction. Defines whether the instruction is an input or an output instruction.
Operands	Lists all the operands of the instruction.
Structured Text	Describes the use of operands in Structured Text format.
Motion Instruction structure	Lists control status bits and values, if any, of the instruction.
Description	Describes the instruction's use. Defines any differences when the instruction is enabled and disabled, if appropriate.
Arithmetic status flags	Defines whether or not the instruction affects arithmetic status flags.
Fault conditions	Defines whether or not the instruction generates minor or major faults. if so, defines the fault type and code.
Error Codes	Lists and defines the applicable error codes.
Status Bits	Lists affected status bits, their states, and definitions.
Example	Provides at least one programming example. Includes a description explaining each example.

Conventions and Related Terms

This manual uses specific terms and conventions.

Set and Clear

This manual uses set and clear to define the status of bits (booleans) and values (non-booleans).

Term	Means
Set	The bit is set to 1 (ON). A value is set to any non-zero number.
Clear	The bit is cleared to 0 (OFF). All the bits in a value are cleared to 0.

An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

Rung Condition

The controller evaluates ladder instructions based on the rung condition preceding the instruction (rung-condition-in). Based on the rung-condition-in and the instruction, the controller sets the rung condition following the instruction (rung-condition-out), which in turn, affects any subsequent instruction.

If the rung-in condition to an input instruction is true, the controller evaluates the instruction and sets the rung-out condition based on the results of the instruction. If the instruction evaluates to true, the rung-out condition is true; if the instruction evaluates to false, the rung-out condition is false.

Sequential Function Chart (SFC)

A Sequential Function Chart is a flowchart that controls your machine or process. SFC uses steps and transitions to perform specific operations or actions. You can use SFC to do the following:

- Organize the functional specification of your system.
- Program and control your system as a series of steps and transitions.

You gain the following advantages by using Sequential Function Chart (SFC):

- Graphical division of processes into major logic pieces
- Faster repeated execution of individual pieces of your logic
- A more simple screen display
- Time to design and debug your program is reduced
- Troubleshooting is faster and easier
- Direct access to the point in the logic where the machine faulted
- Easier to enhance and update

For more detailed information about how to program and use an SFC, see Logix5000 Controllers Sequential Function Charts Programming Manual, publication 1756-PM006.

Instruction Timing

Motion instructions use three types of timing sequences.

Timing Type	Description
Immediate	The instruction completes in one scan.
Message	The instruction completes over several scans because the instruction sends messages to the servo module.
Process	The instruction could take an indefinite amount of time to complete.

Immediate Type Instructions

Immediate type motion instructions execute to completion in one scan. If the controller detects an error during the execution of these instructions, the error status bit sets and the operation ends.

Examples of immediate type instructions include the following:

- Motion Change Dynamics (MCD) instruction
- Motion Group Strobe Position (MGSP) instruction

Immediate instructions work as follows.

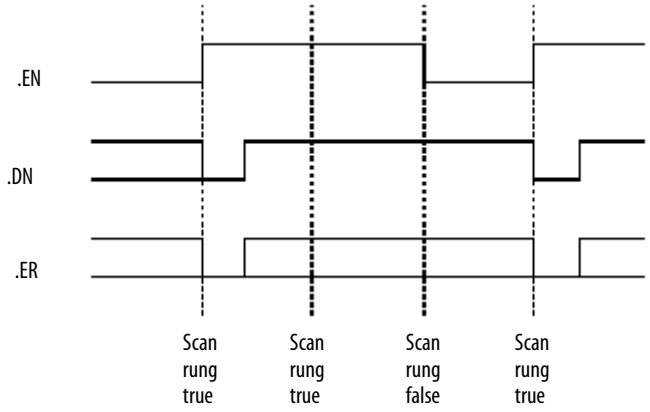
1. When the rung that contains the motion instruction becomes true, the controller does the following:
   - Sets the enable (.EN) bit
   - Clears the done (.DN) bit
   - Clears the error (.ER) bit

   The controller executes the instruction completely.

If the controller	Then
Does not detect an error when the instruction executes	The controller sets the .DN bit.
Detects an error when the instruction executes	The controller sets the .ER bit and stores an error code in the control structure.

2. The next time the rung becomes false after either the .DN or .ER bit sets, the controller clears the .EN bit.
3. The controller can execute the instruction again when the rung becomes true.

Figure 1 - Immediate Type Instructions - Rung Conditions



Message Type Instructions

Message type motion instructions send one or more messages to the servo module.

Examples of message type instructions include the following:

- Motion Direct Drive On (MDO) instruction
- Motion Redefine Position (MRP) instruction

Message type instructions work as follows.

1. When the rung that contains the motion instruction becomes true, the controller does the following:
   - Sets the enable (.EN) bit
   - Clears the done (.DN) bit
   - Clears the error (.ER) bit
2. The controller begins to execute the instruction by setting up a message request to the servo module.

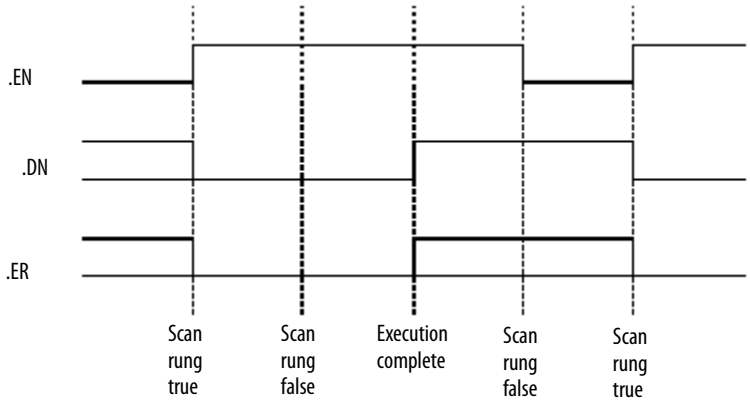
   The remainder of the instruction executes in parallel to the program scan.
3. The controller checks if the servo module is ready to receive a new message.
4. The controller places the results of the check in the message status word of the control structure.
5. When the module is ready, the controller constructs and transmits the message to the module.

   This process can repeat several times if the instruction requires multiple messages.

If the controller	Then
Does not detect an error when the instruction executes	The controller sets the .DN bit if all messaging to the module is completed.
Detects an error when the instruction executes	The controller sets the .ER bit and stores an error code in the control structure.

6. The next time the rung becomes false after either the .DN or .ER bit sets, the controller clears the .EN bit.
7. When the rung becomes true, the controller can execute the instruction again.

Figure 2 - Message Type Instructions - Rung Conditions



Process Type Instructions

Process type motion instructions initiate motion processes that can take an indefinite amount of time to complete.

Examples of process type instructions include the following:

- Motion Arm Watch (MAW) instruction
- Motion Axis Move (MAM) instruction

Process type instructions work as follows.

1. When the rung that contains the motion instruction becomes true, the controller does the following:
   - Sets the enable (.EN) bit
   - Clears the done (.DN) bit
   - Clears the error (.ER) bit
   - Clears the process complete (.PC) bit
   - Sets the in process (.IP) bit

2. The controller initiates the motion process.

If	Then the controller
The controller does not detect an error when the instruction executes	• Sets the .DN bit. • Sets the in process (.IP) bit.
The controller detects an error when the instruction executes	• Sets the .ER bit. • Stores an error code in the control structure.
The controller detects another instance of the motion instruction	Clears the .IP bit for that instance.
The motion process reaches the point where the instruction can be executed again	Sets the .DN bit. For some process type instructions, like MAM, this occurs on the first scan. For others, like MAH, the.DN bit is not set until the entire homing process is complete.
One of the following occurs during the motion process: • The motion process completes • Another instance of the instruction executes • Another instruction stops the motion process • A motion fault stops the motion process	Clears the .IP bit.

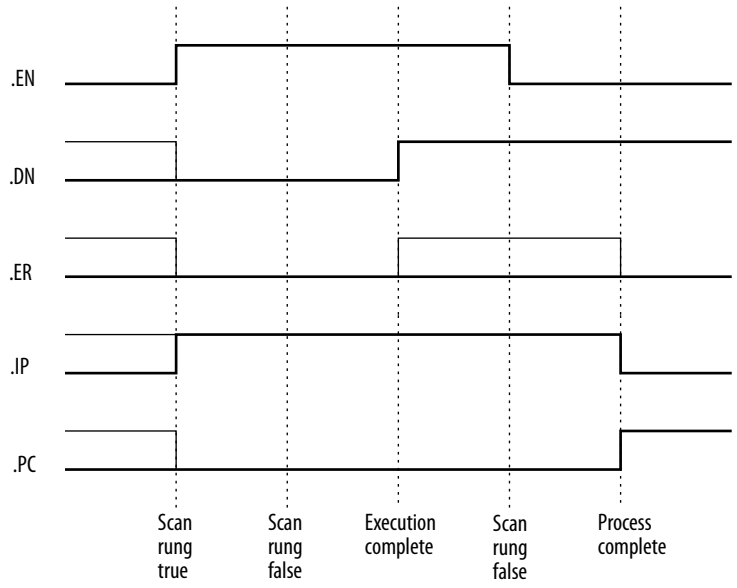
3. After the initiation of the motion process, the program scan can continue.

   The remainder of the instruction and the control process continue in parallel with the program scan.

4. The next time the rung becomes false after either the .DN bit or the .ER bit sets, the controller clears the .EN bit.

5. When the rung becomes true, the instruction can execute again.

Figure 3 - Process Type Instructions - Rung Conditions



Choose a Command

Use this table to choose an instruction and see if it is available as a Motion Direct Command.

Table 2 - Choosing a Motion Direct Command

If You Want To	And	Use This Instruction	Motion Direct Command
Change the state of an axis	Enable the servo drive and activate the axis servo loop.	Motion Servo On (MSO)	Yes
	Disable the servo drive and deactivate the axis servo loop.	Motion Servo Off (MSF)	Yes
	Force an axis into the shutdown state and block any instructions that initiate axis motion.	Motion Axis Shutdown (MASD)	Yes
	Transition an axis to the ready state. If all of the axes of a servo module are removed from the shutdown state as a result of this instruction, the OK relay contacts for the module close.	Motion Axis Shutdown Reset (MASR)	Yes
	Enable the servo drive and set the servo output voltage of an axis.	Motion Direct Drive On (MDO)	Yes
	Disable the servo drive and set the servo output voltage to the output offset voltage.	Motion Direct Drive Off (MDF)	Yes
	Activate the drive control loops for the specified axis and run the motor at the specified speed.	Motion Drive Start (MDS)	Yes
	Clear all motion faults for an axis.	Motion Axis Fault Reset (MAFR)	Yes
Control axis position	Stop any motion process on an axis.	Motion Axis Stop (MAS)	Yes
	Home an axis.	Motion Axis Home (MAH)	Yes
	Jog an axis.	Motion Axis Jog (MAJ)	Yes
	Move an axis to a specific position.	Motion Axis Move (MAM)	Yes
	Start electronic gearing between 2 axes	Motion Axis Gear (MAG)	Yes
	Change the speed, acceleration, or deceleration of a move or a jog that is in progress.	Motion Change Dynamics (MCD)	Yes
	Change the command or actual position of an axis.	Motion Redefine Position (MRP)	Yes
	Calculate a Cam Profile based on an array of cam points.	Motion Calculate Cam Profile (MCCP)	No
	Start electronic camming between 2 axes.	Motion Axis Position Cam (MAPC)	No
	Start electronic camming as a function of time.	Motion Axis Time Cam (MATC)	No
	Calculate the slave value, slope, and derivative of the slope for a cam profile and master value.	Motion Calculate Slave Values (MCSV)	No
Initiate action on all axes	Stop motion of all axes.	Motion Group Stop (MGS)	Yes
	Force all axes into the shutdown state.	Motion Group Shutdown (MGSD)	Yes
	Transition all axes to the ready state.	Motion Group Shutdown Reset (MGSR)	Yes
	Latch the current command and actual position of all axes.	Motion Group Strobe Position (MGSP)	Yes
Arm and disarm special event checking functions such as registration and watch position	Arm the watch-position event checking for an axis.	Motion Arm Watch (MAW)	Yes
	Disarm the watch-position event checking for an axis.	Motion Disarm Watch (MDW)	Yes
	Arm the servo-module registration-event checking for an axis.	Motion Arm Registration (MAR)	Yes
	Disarm the servo-module registration-event checking for an axis.	Motion Disarm Registration (MDR)	Yes
	Arm an output cam for an axis and output.	Motion Arm Output Cam (MAOC)	No
	Disarm one or all output cams connected to an axis.	Motion Disarm Output Cam (MDOC)	No

Table 2 - Choosing a Motion Direct Command (Continued)

If You Want To	And	Use This Instruction	Motion Direct Command
Tune an axis and run diagnostic tests for your control system. These tests include: • Motor/encoder hookup test • Encoder hookup test • Marker test	Use the results of an MAAT instruction to calculate and update the servo gains and dynamic limits of an axis.	Motion Apply Axis Tuning (MAAT)	No
	Run a tuning motion profile for an axis	Motion Run Axis Tuning (MRAT)	No
	Use the results of an MRHD instruction to set encoder and servo polarities.	Motion Apply Hookup Diagnostics (MAHD)	No
	Run one of the diagnostic tests on an axis.	Motion Run Hookup Diagnostics (MRHD)	No
Control multi-axis coordinated motion(1)	Start a linear coordinated move for the axes of coordinate system.	Motion Coordinated Linear Move (MCLM)	No
	Start a circular move for the for the axes of coordinate system.	Motion Coordinated Circular Move (MCCM)	No
	Change in path dynamics for the active motion on a coordinate system.	Motion Coordinated Change Dynamics (MCCD)	No
	Stop the axes of a coordinate system or cancel a transform.	Motion Coordinated Stop (MCS)	No
	Shutdown the axes of a coordinate system.	Motion Coordinated Shutdown (MCSD)	No
	Start a transform that links two coordinate systems together. This is like bidirectional gearing.	Motion Coordinated Transform (MCT)	No
	Calculate the position of one coordinate system with respect to another coordinate system.	Motion Calculate Transform Position (MCTP)	No
	Transition the axes of a coordinate system to the ready state and clear the axis faults.	Motion Coordinated Shutdown Reset (MCSR)	No

(1) For more information on these Motion Coordinated Instructions, see Motion Coordinate System User Manual, publication MOTION-UM002.

Additional Resources

These documents contain additional information concerning related products from Rockwell Automation.

Table 3 - Additional Resources

Resource	Description
1768 CompactLogix Controllers User Manual, publication 1768-UM001	Describes the necessary tasks to install, configure, program, and operate a 1768 CompactLogix system.
ControlLogix System User Manual, publication 1756-UM001	Describes the necessary tasks to install, configure, program, and operate a ControlLogix system.
GuardLogix Controller Systems Safety Reference Manual, publication 1756-RM093	Contains detailed requirements for achieving and maintaining SIL 3 with the GuardLogix controller system.
GuardLogix Controllers User Manual, publication 1756-UM020	Provides information on configuring and programming the 1756 GuardLogix controller.
Integrated Motion on the Ethernet/IP Network Configuration and Startup Reference Manual, publication MOTION-UM003	Describes how to configure an integrated motion application and to start up your motion solution by using the ControlLogix system.
Integrated Motion on the Ethernet/IP Network Reference Manual, publication MOTION-RM003	Provides a programmer with details about the Integrated Motion Control Modes, Control Methods, and AXIS_CIP_DRIVE attributes.
Integrated Architecture and CIP Sync Configuration Application Technique, publication IA-AT003	Provides detailed configuration information on CIP Sync technology and time synchronization.
Kinetix® 6200 and Kinetix 6500 Modular Multi-axis Servo Drives User Manual, publication 2094-UM002	Provides information on installing, configuring, starting, troubleshooting, and applications for the Kinetix 6200 and Kinetix 6500 servo drive systems.
Kinetix 350 Single-axis EtherNet/IP Servo Drives User Manual, publication 2097-UM002	Provides detailed information on wiring, applying power, troubleshooting, and integration with ControlLogix, or CompactLogix controller platforms.
Kinetix 6000 Multi-axis Servo Drives User Manual, publication 2094-UM001	Provides detailed information on wiring, applying power, troubleshooting, and integration with ControlLogix, CompactLogix, or SoftLogix controller platforms.
Logix5000 Controllers Common Procedures, publication 1756-PM001	Guides the development of projects for Logix5000 controllers. It provides resource links to individual publications.
Motion Coordinate System User Manual, publication MOTION-UM002	Provides instructions on creating a motion coordinate system.
PowerFlex 750-Series AC Drives Programming Manual, publication 750-PM001	Provides information needed to install, start, and troubleshoot PowerFlex 750-Series Adjustable Frequency AC Drives.
PowerFlex 755 Drive Embedded EtherNet/IP Adapter User Manual, publication 750COM-UM001	Provides information on installing, configuring, staring, troubleshooting, and applications for the PowerFlex 755 Drive Embedded EtherNet/IP Adapter.
SERCOS and Analog Motion Configuration and Setup User Manual, publication MOTION-UM001	Describes how to configure an integrated motion application and to start your motion solution by using SERCOS or an analog control solution.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, http://www.rockwellautomation.com/global/certification/overview.page	Provides declarations of conformity, certificates, and other certification details.

You can view or download publications at http://www.rockwellautomation.com/literature.

To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Notes:

Motion State Instructions

(MSO, MSF, MASD, MASR, MDO, MDF, MDS, MAFR)

ATTENTION: Tags used for the motion control attribute of instructions should only be used once. Reuse of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

Motion state control instructions directly control or change the operating states of an axis. These are the motion state instructions.

Table 4 - Choosing a Motion State Instruction

If you want to	Use this instruction	Page	Available in these languages
Enable the servo drive and activate the axis servo loop.	Motion Servo On (MSO)	39	Relay ladder Structured text
Disable the servo drive and deactivate the axis servo loop.	Motion Servo Off (MSF)	43	
Force an axis into the shutdown operating state. Once the axis is in the shutdown operating state, the controller will block any instructions that initiate axis motion.	Motion Axis Shutdown (MASD)	48	
Change an axis from an existing shutdown operating state to an axis ready operating state. If all of the axes of a servo module are removed from the shutdown state as a result of this instruction, the OK relay contacts for the module will close.	Motion Axis Shutdown Reset (MASR)	54	
Enable the servo drive and set the servo output voltage of an axis.	Motion Direct Drive On (MDO)	59	
Deactivate the servo drive and set the servo output voltage to the output offset voltage.	Motion Direct Drive Off (MDF)	65	
Activate the drive control loops for the specified axis and run the motor at the specified speed.	Motion Drive Start (MDS)	69	
Clear all motion faults for an axis.	Motion Axis Fault Reset (MAFR)	75	

These are the five operating states of a non-CIP axis.

Table 5 - Operating States of a Non-CIP Axis

Operating State	Description
Axis Ready	This is the normal power-up state of the axis. In this state: • the servo module drive enable output is inactive. • servo action is disabled. • no servo faults are present.
Direct Drive Control	This operating state allows the servo module DAC to directly control an external drive. In this state: • the servo module drive enable output is active. • position servo action is disabled.
Servo Control	This operating state allows the servo module to perform closed loop motion. In this state: • the servo module drive enable output is active. • servo action is enabled. • the axis is forced to maintain the commanded servo position.
Axis Faulted	In this operating state, a servo fault is present, and the status of the drive enable output, the action of the servo, and the condition of the OK contact depend on the faults and fault actions that are present.
Shutdown	This operating state allows the OK relay contacts to open a set of contacts in the E-string of the drive power supply. In this state: • the servo module drive enable output is inactive. • servo action is disabled. • the OK contact is open.

These are the 16 operating states of a CIP axis.

Table 6 - Operating States of a CIP Axis

Operating State	In RSLogix 5000® programming software	Description
Initializing	0	During the Initializing State, the drive first initializes all attributes to their factory default values, that is, resets all active faults. The drive then waits for the controller to establish connections to it. Once connections are established, the controller sets configuration attributes in the drive to values stored in the controller. If the drive supports synchronous operation, the controller then synchronizes with the drive. Once this process has been completed successfully, the drive and all its associated axis instances transition to the Pre-charge state. If a problem is found during the initializing process, an Initialization Fault is generated. An Initialization Fault is an unrecoverable fault. You can only clear the fault can via a power cycle or a drive reset. If the connection to the drive closes for any reason during operation, the drive returns to the Initializing State.
Pre-charge	1	The drive is waiting for the DC Bus to fully charge, that is, the DC Bus Up status bit is cleared. Once the DC Bus reaches an operational voltage level, that is, DC Bus Up status bit is set, the axis transitions to the Stopped state. The drive's power structure is always disabled in this state, that is, the Power Structure Enabled status bit is cleared. Any attempt to enable the drive via the Axis Control mechanism while in this state is reported back to the controller as an error in the Response Status and the axis remains in the Pre-charge state.
Stopped	2	In the Stopped state, the drive's inverter power structure should either be disabled and free of torque, that is, the Power Structure Enabled status bit is cleared, or held in a static condition via an active control loop, that is, Power Structure Enabled status bit is set. The drive cannot initiate motion in the Stopped state nor can the drive respond to a planner generated command reference, that is, the Tracking Command status bit is cleared. In general, the axis should be at rest. However, if you apply an external force or torque to the load, a brake can be needed to maintain the rest condition. In the Stopped state, main power is applied to the drive and the DC Bus are at an operational voltage level. If there are any Start Inhibited conditions detected while in this state, the axis transitions to the Start Inhibited state. If an Enable request or one of the Run Test service requests is applied to an axis in the Stopped state, the motion axis transitions to the Starting state.
Starting	3	When an Enable request is given to an axis in the Stopped or Stopping state while it is performing a Flying Start, the axis immediately transitions to the Starting state. In this state, the drive checks the following conditions before it transitions to the Running state. • Brake Release delay time • Induction Motor flux level The drive control and power structures are activated during the Starting state, that is, the Power Structure Enabled status bit is set. But the command reference is set to a local static value and does not track the command reference derived from the motion planner, that is, the Tracking Command status bit is cleared. If all the starting conditions are met, the axis state transitions to either the Running state or the Testing state.
Running	4	The drive's power structure is active, that is, the Power Structure Enabled status bit is set. Additionally, the selected Control Mode is enabled and actively tracking command data from the controller-based or drive-based motion planner output to affect axis motion, that is, the Tracking Command status bit is set.
Testing	5	When any one of the Run Test request services is sent to the motion axis while in the Stopped state, that is, services that require an active power structure to execute, the axis immediately transitions to the Starting state, that is, the Power Structure Enabled status bit is set. Then once the Starting state conditions are met, the axis transitions to the Testing state. Like the Running state, in the Testing state, the drive's power structure is active. The motion axis remains in this state for the duration of the requested test procedure and then returns to the Stopped state. The motion axis can also exit the Testing state by either a fault or an explicit Axis Control request.
Stopping	6	When a Disable request is issued to an axis in the Running or Testing state, the axis immediately transitions to the Stopping state. In this state, the axis is in the process of stopping and no longer tracks command data from the motion planner, that is, the Tracking Command status bit is cleared. Once the selected Stopping Mode procedure has completed, the axis transitions to the Stopped state.
Aborting	7	When a Major Fault occurs in the drive while the axis is in either the Running or Testing states, the motion axis immediately transitions to the Aborting state. In this state, the axis is in the process of stopping and no longer tracks command data from the motion planner, that is, the Tracking Command status bit is cleared. The Aborting state executes the appropriate stopping action as specified by the drive. As with the Stopping state, in the Aborting state the power structure remains active, that is, the Power Structure Enabled status bit is set, for as long as the stopping action takes to complete. Once the stopping procedure is complete the axis transitions to the Faulted state. When faults conditions are detected in the controller that are not visible to the drive, or when the drive reports a Minor Fault condition, the controller brings the axis to a stop, either directly via an Axis Control state change request or motion planner stop, or indirectly via a fault handler in the user program. If the Axis State reported by the drive is Stopping, the controller sets the CIP Axis State to Aborting based on the presence of the fault condition.

Table 6 - Operating States of a CIP Axis

Operating State	In RSLogix 5000® programming software	Description
Faulted	8	The faulted state is identical to the Stopped state or the Shutdown state with the exception that there are one or more faults active. Faults are latched conditions. Therefore, a Fault Reset is required to clear the faults and, assuming the original fault condition has been removed, the axis transitions to the Axis State of the drive. **Fault** / **Description** CIP Initialization Faults — Faults that occur when the drive transitions out of the Initializing state. These faults can apply to a specific axis or the entire drive. CIP Axis Faults — Faults that apply to a specific axis and are the direct result of Axis Exceptions configured to generate a Fault response. Axis exceptions are run-time conditions that are related to Motor, Inverter, Converter, Bus Regulator, and Feedback components. Safety Fault — Faults that apply to a specific axis and are generated by a fault condition detected in the drive's safety monitor functionality. A Safety Fault always results in the axis transitioning to the Stopped state. Motion Fault — Faults generally associated with fault conditions generated by the motion planner function. These faults can include conditions related to the input, for example, actual position, or output, for example, command position, signals. Module Fault — Faults that apply to the entire drive and affect all axes associated with that drive. Module faults include all node faults reported by the drive and also communication fault conditions detected on the controller side of the motion connection. Group Fault — Faults related to the motion group object function and affect all axes associated with the motion group. Group Fault conditions are detected by controller and are associated with the time synchronization function common to all axes in the motion group. Configuration Fault — Fault generated anytime there is an error when sending configuration data to the drive.
Start Inhibited	9	This state is the same as the Stopped state with the exception that the axis has one or more "start inhibit" conditions that prevent it from successfully transitioning to the Starting state. Once corrected, the axis state automatically transitions back to the Stopped state.
Shutdown	10	When a Shutdown request is issued to the drive or a Shutdown fault action is executed by the drive, the targeted axis immediately transitions to the Shutdown state. The Shutdown state has the same basic characteristics of the Stopped state except that it can be configured via the Shutdown Action attribute to drop the DC Bus power to the drive's power structure. Regardless of whether DC Bus power is disconnected, this state requires an explicit Shutdown Reset request from the controller to transition to the Pre-charge state. If the drive is configured to keep DC Bus power active while in the Shutdown state, then the motion axis transitions through the Pre-charge state to the Stopped state. In the case where a Shutdown fault action is initiated by the drive in response to an exception condition that is configured to be a major fault, the drive executes the Shutdown action. However, the axis goes to the Faulted state, not the Shutdown state. Similarly, when the axis is in the Shutdown state and a major fault condition occurs, the axis transitions to the Faulted state. A Fault Reset request from the controller clears the fault and, assuming the original fault condition has been removed, the axis transitions to the Shutdown state. A Shutdown Reset request from the controller, however, both clears the fault and performs a shutdown reset so, assuming the original fault condition has been removed, the axis transitions to the Pre-charge state.
Axis Inhibited	11	If you inhibit the axis, the associated instance in the CIP Motion connection is eliminated and the axis state transitions to the Axis Inhibited state. If this is the only instance supported by the CIP Motion connection, the connection itself is closed. The Axis Inhibited state is a controller-only sub state of the Self-test state(1). The Axis Inhibited condition is checked during the controller Self-test state as qualification for transition to the Initializing state. If currently Axis Inhibited, you must execute an Un-Inhibit operation to transition to the Initializing state and restore axis function.
Not Grouped	12	If a CIP Motion axis is created and not associated with a Motion Group, the axis state is set to the Not Grouped state. A CIP Motion axis must be assigned to a Motion Group for the axis to be updated by the periodic Motion Task and carry out its function. This condition is checked during the controller Self-test state as qualification for transition to the Initializing state. For this reason, the Not Grouped state is considered a controller-only sub state of the Self-test state.
No Device	13	If the CIP Motion axis in the controller is created, but not currently associated with a drive, the axis state indicates the No Device state. A CIP Motion axis must be associated with a physical drive to function. This condition is checked during the controller Self-test state as qualification for transition to the Initializing state. For this reason, the No Device state is considered a controller-only sub state of the Self Test state.

(1) The Self-test state is a drive state. This state does not appear in Logix Designer application as an operating state of a CIP axis. Instead, self-test is represented as the Initializing state for a CIP axis.

Motion Servo On (MSO)

Use the Motion Servo On (MSO) instruction to activate the drive amplifier for the specified axis and to activate the axis' servo control loop.

Operands

The MSO instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

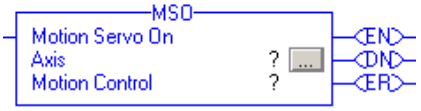


Table 7 - MSO Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MSO(Axis,MotionControl);

The operands are the same as those for the relay ladder MSO instruction.

MOTION_INSTRUCTION Structure

Table 8 - MSO Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' servo action has been successfully enabled and the drive enable and servo active status bits have been set.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MSO instruction directly activates the drive and enables the configured servo loops associated with a physical servo axis. It can be used anywhere in a program, but should not be used while the axis is moving. If this is attempted, the MSO instruction generates an 'Axis in Motion' error.

The MSO instruction automatically enables the specified axis by activating the drive and by activating the associated servo loop. With a non-CIP axis, the resulting state of the axis is referred to as the Servo Control state. With a CIP axis, the resulting state of the axis is referred to as the Running state.

The most common use of this instruction is to activate the servo loop for the specified axis in its current position in preparation for commanding motion.

Executing the Instruction

To successfully execute a MSO instruction, the targeted axis must be configured as a Servo axis. If this condition is not met the instruction errors.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

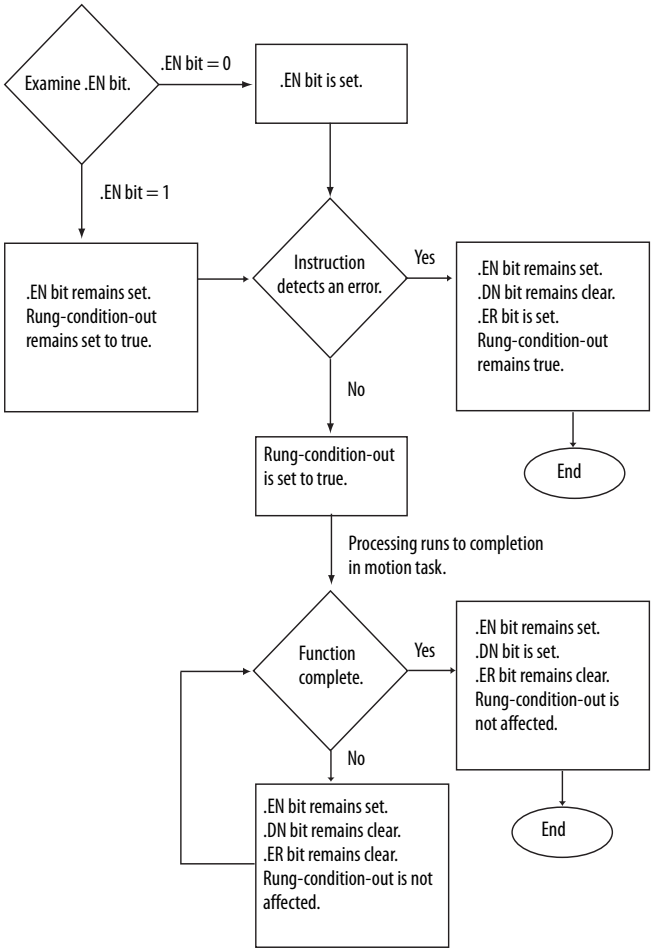
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MSO Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MSO Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MSO instruction receives a Servo Message Failure (12) error message.

Table 9 - MSO Extended Error Codes

Extended Error Code (decimal)	Associated Error Code (decimal)	Meaning
Object Mode conflict (12)	SERVO_MESSAGE_FAILURE (12)	Axis is in shutdown.
Permission Denied (15)	SERVO_MESSAGE_FAILURE (12)	Enable input switch error. (sercos)
Device in wrong state (16)	SERVO_MESSAGE_FAILURE (12)	Device state not correct for action. (sercos)

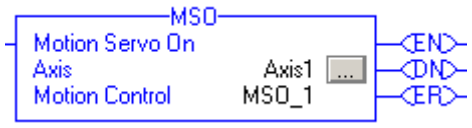
MSO Changes to Status Bits

Bit Name	State	Meaning
ServoActStatus	TRUE	Axis is in Servo Control state with the servo loop active.
DriveEnableStatus	TRUE	The axis drive enable output is active.

Example

When the input conditions are true, the controller enables the servo drive and activates the axis servo loop configured by **axis1**.

Relay Ladder



Structured Text

```
MSO(Axis0,MSO_1);
```

Motion Servo Off (MSF)

Use the Motion Servo Off (MSF) instruction to deactivate the drive output for the specified axis and to deactivate the axis' servo loop.

IMPORTANT	If you execute an MSF instruction while the axis is moving, the axis coasts to an uncontrolled stop.

Operands

The MSF instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 10 - MSF Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform action upon.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MSF(Axis,MotionControl);

The operands are the same as those for the relay ladder MSF instruction.

MOTION_INSTRUCTION Structure

Table 11 - MSF Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' servo action been successfully disabled and the drive enable and servo active status bits have both been cleared.
.ER (Done) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MSF instruction directly and immediately turns off drive output and disables the servo loop on any physical servo axis. With non-CIP motion, this places the axis in the Axis Ready state. With integrated motion, this places the axis in the Stopped state. The MSF instruction also disables any motion planners that can be active at the time of execution. The MSF instruction requires no parameters – simply enter or select the desired axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

You can use the MSF instruction to turn servo action off when you must move the axis by hand. Because the position continues to be tracked even with servo action OFF. When the servo loop is turned ON again, by the Motion Servo On (MSO) instruction, the axis is again under closed-loop control, at the new position.

IMPORTANT When drive faults with a stopping action of Current Decel & Hold and the fault is cleared, the drive display shows that it is STOPPED but the drive is still enabled (Bus LED is solid) and the drive is still holding position.

The drive cannot be disabled with an MSF command and MASR has no effect. The only way to disable the drive is to issue an MASD command or to change the stopping action to Disable and Coast.

The axis stopping behavior varies depending upon the type of drive. In some cases the axis coasts to a stop and in other cases the axis decelerates to a stop by using the drive's available stopping torque.

Executing the Instruction

To execute an MSF instruction successfully, the targeted axis must be configured as a Servo axis. If this condition is not met, the instruction errs. If you have an Axis Type of Virtual the instructions errors because with a Virtual Axis the servo action and drive enable status are forced to always be true. A Consumed axis data type also errors because only the producing controller can change the state of a consumed axis.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

Additionally, the MSF instruction supports canceling the Motion Drive Start (MDS) instruction. This includes clearing the MDS In Process (.IP) bit, and clearing the DirectVelocityControlStatus and the DirectTorqueControlStatus bit in the Motion Status attribute.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

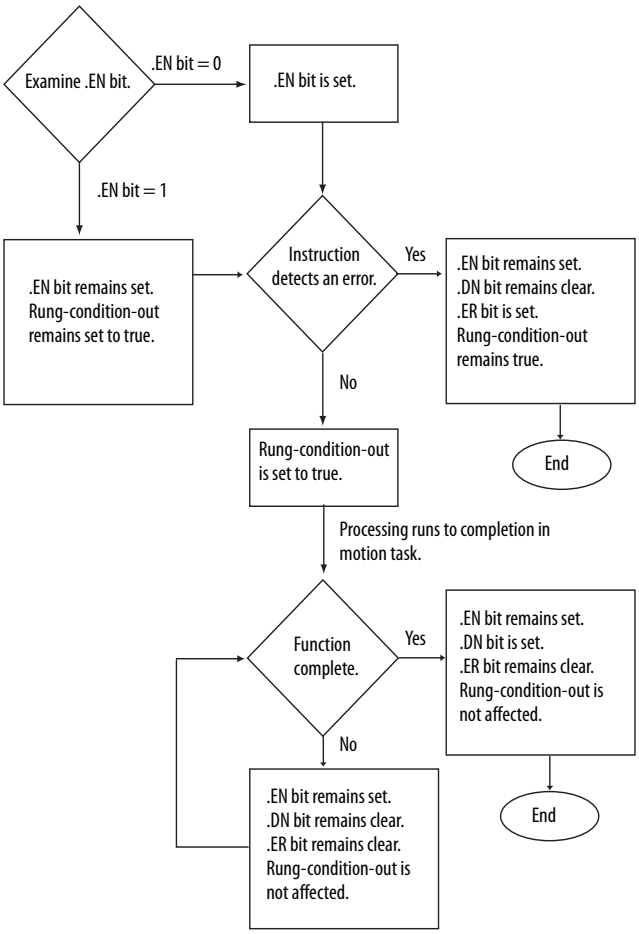
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MSF Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MSF Changes to Status Bits

Bit Name	State	Meaning
ServoActionStatus	FALSE	Axis is in Servo On state with the servo loop active.
DecelStatus	FALSE	Axis Drive Enable output is active.

Motion Status Bits

Table 12 - MSF Motion Status Bits

Bit Name	State	Meaning
AccelStatus	FALSE	Axis is not Accelerating.
DecelStatus	FALSE	Axis is not Decelerating.
MoveStatus	FALSE	Axis is not Moving.
JogStatus	FALSE	Axis is not Jogging.
GearingStatus	FALSE	Axis is not Gearing.
HomingStatus	FALSE	Axis is not Homing.
StoppingStatus	FALSE	Axis is not Stopping.
PositionCamStatus	FALSE	Axis is not Position Camming.
TimeCamStatus	FALSE	Axis is not Time Camming.
PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.
TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
GearingLockStatus	FALSE	Axis is not in a Gear Locked condition.
PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition.
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

MSF Example

When the input conditions are true, the controller disables the servo drive and the axis servo loop configured by Axis0.

Relay Ladder

Structured Text

```
MSF(Axis0,MSF_1);
```

Motion Axis Shutdown (MASD)

Use the Motion Axis Shutdown (MASD) instruction to force a specified axis into the Shutdown state. The Shutdown state of an axis is the condition where the drive output is disabled, servo loop deactivated, and any available or associated OK solid-state relay contacts open. The axis remains in the Shutdown state until either an Axis or Group Shutdown Reset is executed.

Operands

The MASD instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 13 - MASD Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	The name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MASD(Axis,MotionControl);

The operands are the same as those for the relay ladder MASD instruction.

MOTION_INSTRUCTION Structure

Table 14 - MASD Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis have been successfully set to Shutdown state.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MASD instruction directly and immediately disables drive output, disables the servo loop, and opens any associated OK contacts. This action places the axis into the Shutdown state.

Another action initiated by the MASD instruction is the clearing of all motion processes in progress and the clearing of all the motion status bits. Associated with this action, the command also clears all motion instruction IP bits that are currently set for the targeted axis.

The MASD instruction forces the targeted axis into the Shutdown state. One of the unique characteristics of the Shutdown state is that, when available, the OK solid state relay contact for the motion module or drive is Open. This feature can be used to open up the E-Stop string that controls main power to the drive system. Note that there is typically only one OK contact per motion module which means that execution of an MASD instruction for either axis associated with a given module opens the OK contact.

Another characteristic of the Shutdown state is that any instruction that initiates axis motion is blocked from execution. Attempts to do so result in an execution error. Only by executing one of the Shutdown Reset instructions can motion be successfully initiated.

Executing the Instruction

To successfully execute an MASD instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. If not, the instruction errs.

The axis remains in the shutdown state until either a Motion Axis Shutdown Reset (MASR) instruction or a Motion Group Shutdown Reset (MGSR) instruction executes.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

Additionally, the MASD instruction supports canceling the Motion Drive Start (MDS) instruction. This includes clearing the MDS In Process (.IP) bit, and clearing the DirectVelocityControlStatus and the DirectTorqueControlStatus bit in the Motion Status attribute.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

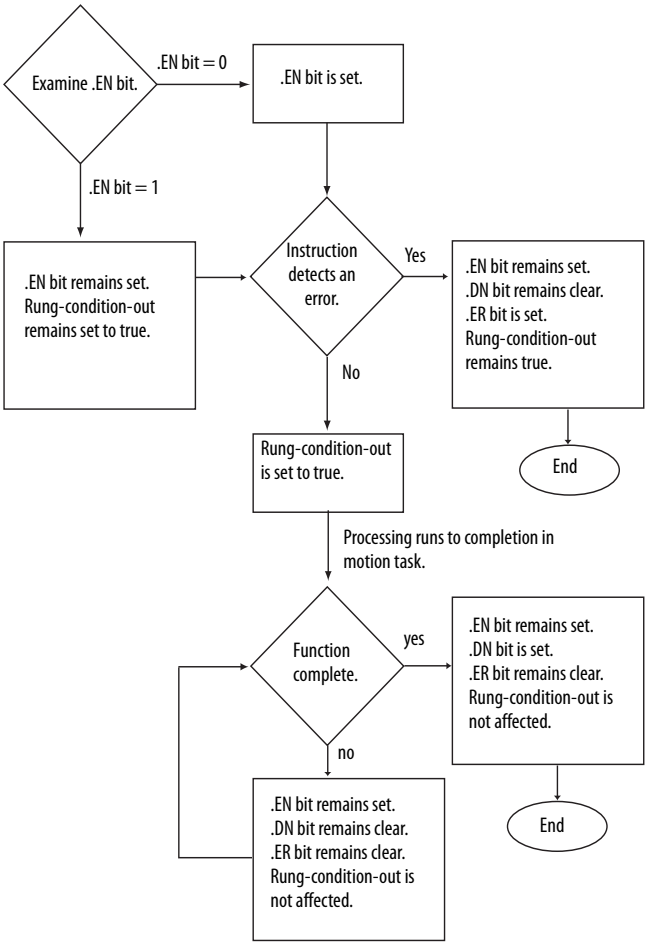
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MASD Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MASD Changes to Status Bits

Bit Name	State	Meaning
ServoActStatus	FALSE	• The axis is in the axis ready state. • The servo loop is inactive.
DriveEnableStatus	FALSE	The drive enable output is inactive.
ShutdownStatus	TRUE	The axis is in the shutdown state.

Motion Status Bits

Table 15 - MASD Motion Status Bits

Bit Name	State	Meaning
AccelStatus	FALSE	Axis is not Accelerating
DecelStatus	FALSE	Axis is not Decelerating
MoveStatus	FALSE	Axis is not Moving
JogStatus	FALSE	Axis is not Jogging
GearingStatus	FALSE	Axis is not Gearing
HomingStatus	FALSE	Axis is not Homing
StoppingStatus	FALSE	Axis is not Stopping
PositionCamStatus	FALSE	Axis is not Position Camming
TimeCamStatus	FALSE	Axis is not Time Camming
PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.
TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
GearingLockStatus	FALSE	Axis is not in a Gear Locked condition
PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

MASD Example

When the input conditions are true, the controller forces *axis1* into the shutdown operating state.

Relay Ladder

Structured Text

```
MASD(Axis0,MASD_1);
```

Motion Axis Shutdown Reset (MASR)

Use the Motion Axis Shutdown (MASR) instruction to transition an axis from an existing Shutdown state to an Axis Ready state. All faults associated with the specified axis are automatically cleared. If, as a result of this instruction, all axes of the associated motion module are no longer in the Shutdown condition, the OK relay contacts for the module close.

Operands

The MASR instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 16 - MASR Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MASR(Axis,MotionControl);

The operands are the same as those for the relay ladder MASR instruction.

MOTION_INSTRUCTION Structure

Table 17 - MASR Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis is successfully reset from Shutdown state.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MASR instruction clears all axis faults and takes the specified axis out of the Shutdown state. If the motion module supports an OK contact, and no other module axis is in the Shutdown state, the MASR instruction results in closure of the module's OK solid-state relay contact. Regardless of the OK contact condition, execution of the MASR places the axis into the Axis Ready state.

Just as the Motion Axis Shutdown (MASD) instruction forces the targeted axis into the Shutdown state, the MASR instruction takes the axis out of the Shutdown state into the Axis Ready state. One of the unique characteristics of the Shutdown state is that any associated OK solid state relay contact for the motion module is Open. If, as a result of an MASR instruction there are no axes associated with a given motion module in the Shutdown state, the OK relay contacts close as a result of the MASR. This feature can be used to close the E-Stop string that controls main power to the drive system and, thus, permit the customer to reapply power to the drive. Note that there is typically only one OK contact per motion module which means that execution of the MASR instruction can be required for all axes associated with a given module for the OK contact to close.

To successfully execute a MASR instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errs.

The MASR instruction is a procedure type command that is processed from the Logix controller, through the sercos module, and to the associated drives. For applications using a Logix controller with the Logix Designer application, version 13 and earlier, the .DN bit of the instruction was set when the sercos module acknowledged the procedure request. The drives acknowledged the command and processed it quickly. To the user, the .DN bit operation appeared to indicate the successful completion of clearing faults almost immediately.

For applications using a Logix controller with the Logix Designer application, version 15 and later, current drive firmware versions have incorporated a new portion of the fault clearing mechanism that triggers a position recovery process in the drive that recalculates commutation. This recalculation could result in the drive not in a ready state to go servo on for 0.5 to 5 seconds after the reset was performed. To reflect this change in the instruction, the .DN bit operation was changed to wait for the drive to complete the reset procedure. The end result is that you will not see a successful completion of clearing faults almost immediately as you can have in previous versions.

Executing the Instruction

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

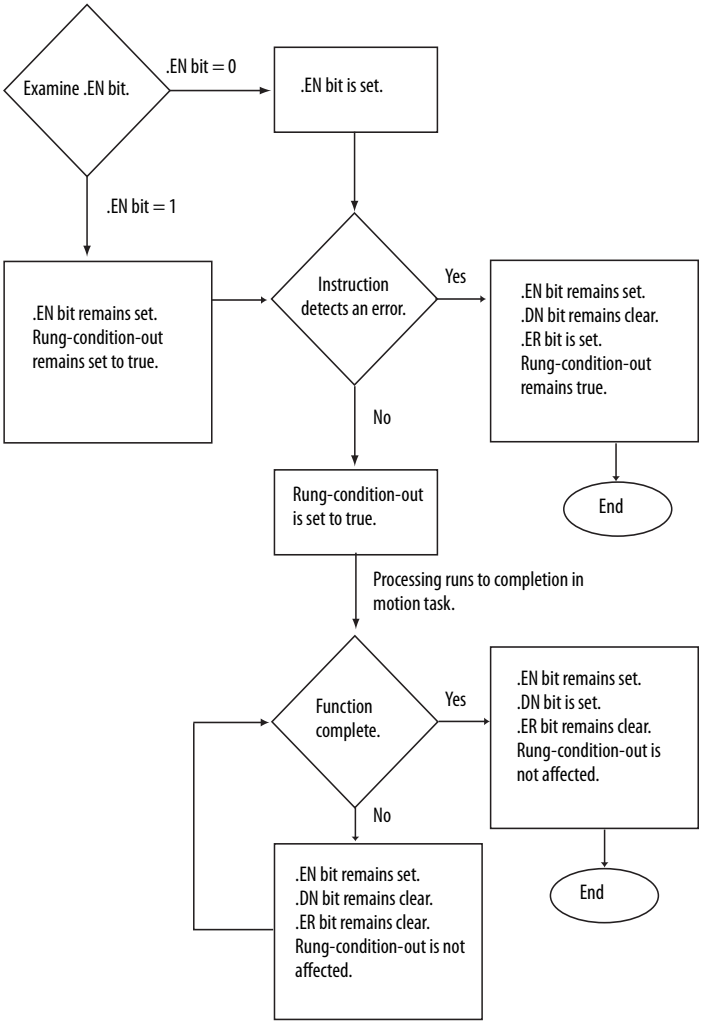
Fault Conditions

There are no fault conditions.

MASR Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A

Condition	Ladder Diagram Action	Structured Text Action
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

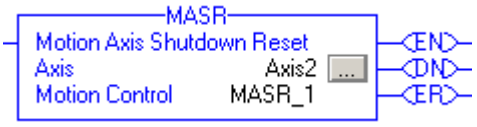
MASR Changes to Status Bits

Bit Name	State	Meaning
ShutdownStatus	FALSE	The axis is not in the shutdown state.

MASR Example

When the input conditions are true, the controller resets *axis1* from a previous shutdown operating state into an axis ready operating state.

Relay Ladder



Structured Text

```
MASR(Axis0,MASR_1);
```

Motion Direct Drive On (MDO)

Use the Motion Direct Drive On (MDO) instruction in conjunction with motion modules that support an external analog servo drive interface, for example the 1756–M02AE or 1784-PM02AE servo module. This instruction activates the module's Drive Enable, enabling the external servo drive, and also sets the servo module's output voltage of the drive to the specified voltage level. The value for Drive Output can be specified in Volts or % of maximum axis' Output Limit.

Operands

The MDO instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

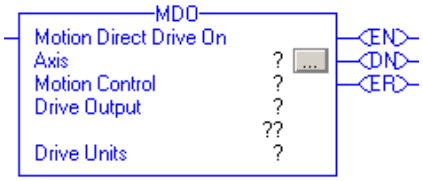


Table 18 - MDO Relay Ladder Descriptions

Operand	Data Type	Description
Axis	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION Tag	Structure used to access instruction status parameters.
Drive Output	REAL	Voltage to output in % of servo Output Limit or in Volts
Drive Units	Boolean	Units in which the Drive Output value is interpreted.

Structured Text

MDO(Axis,MotionControl,DriveOutput,DriveUnits);

The operands are the same as those for the relay ladder MDO instruction.

Enter your selection for the operands that require you to select from available options.

This Operand	Has These Options Which You...	
	Enter as Text	Or Enter as a Number
DriveUnits	Volts Percent	0 1

MOTION_INSTRUCTION Structure

Table 19 - MDO Motion_Instruction Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' drive enable bit is activated and the specified analog output is successfully applied.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you entered a Drive Output value that was too large.

Description

For motion modules with an external servo drive interface, like the 1756-M02AE or 1784-PM02AE, the MDO instruction can be used to directly enable the Drive Enable output of the axis and set the analog output to a specified level determined by the Drive Output parameter. The Drive Output parameter can be expressed as a voltage, or as a percent of the maximum configured output voltage value given by the Output Limit attribute.

The MDO instruction can only be used on a physical axis whose Axis Type is configured for Servo. The instruction only executes when the axis' is in the Axis Ready state (for example, servo action is OFF). The resulting state of the axis is referred to as the Drive Control state.

The MDO instruction automatically enables the specified axis by activating the appropriate Drive Enable output before setting the servo module's analog output to the specified voltage value. There is, typically, a 500 msec delay between the activation of the drive enable output and the setting of the analog output to the specified level to allow the drive's power structure to stabilize. To minimize drift during this drive enabling delay, the output voltage to the drive is set to the Output Offset attribute value (default is zero). Thereafter the output voltage is given by the specified Drive Output value of the MDO instruction and indicated by the Servo Output status attribute value.

The 16-bit DAC hardware associated with various Logix servo modules limits the effective resolution of the Direct Drive Motion Control to 305 μV or 0.003%. In the case of Direct Drive operation, the module's servo loop is inactive and bypassed. The Motion Direct Drive On instruction is only affected by the Servo Output Polarity configuration bit, the Output Offset, and Output Limit attributes for the axis. In the case where Output Limit configuration value is reduced below the current output voltage value, the Servo Output value is automatically clamped to the Output Limit value.

The most common use of this instruction is to provide an independent programmable analog output as an open loop speed reference for an external drive or for testing an external servo drive for closed loop operation.

Executing the Instruction

To successfully execute a MDO instruction, the targeted axis must be configured as a Servo axis and be in the Axis Ready state, with servo action off. If these conditions are not met the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Loss of Feedback When Using an MDO Instruction

If you experience a loss of feedback when issuing an MDO instruction and need to move the axis with an MDO instruction, follow these steps:

1. Set the Feedback Fault Actions to Status Only.
2. When a feedback fault occurs, issue an MSF instruction to turn the servo off.
3. Issue an MAFR instruction to clear the feedback fault status.

The MDO instruction executes without another feedback fault shutting down the system. However, the feedback fault status remains feedback fault condition exists.

IMPORTANT Keep in mind the following when using the previous steps:

- Once feedback has been lost, the reported position can not be valid. To reestablish a valid position, perform another home operation.
- The steps listed previously are only valid for the following modules:
  - 1756-M02AE module in an Logix Designer project, versions 15 to 17
  - 1756-M02AS module in an Logix Designer project, version 15
- If you need to issue another MDO instruction, such as to modify the output voltage, when the first MDO instruction is executing, you must first issue an MSF instruction to stop the servo and then issue an MAFR instruction to clear the feedback fault.

Arithmetic Status Flags

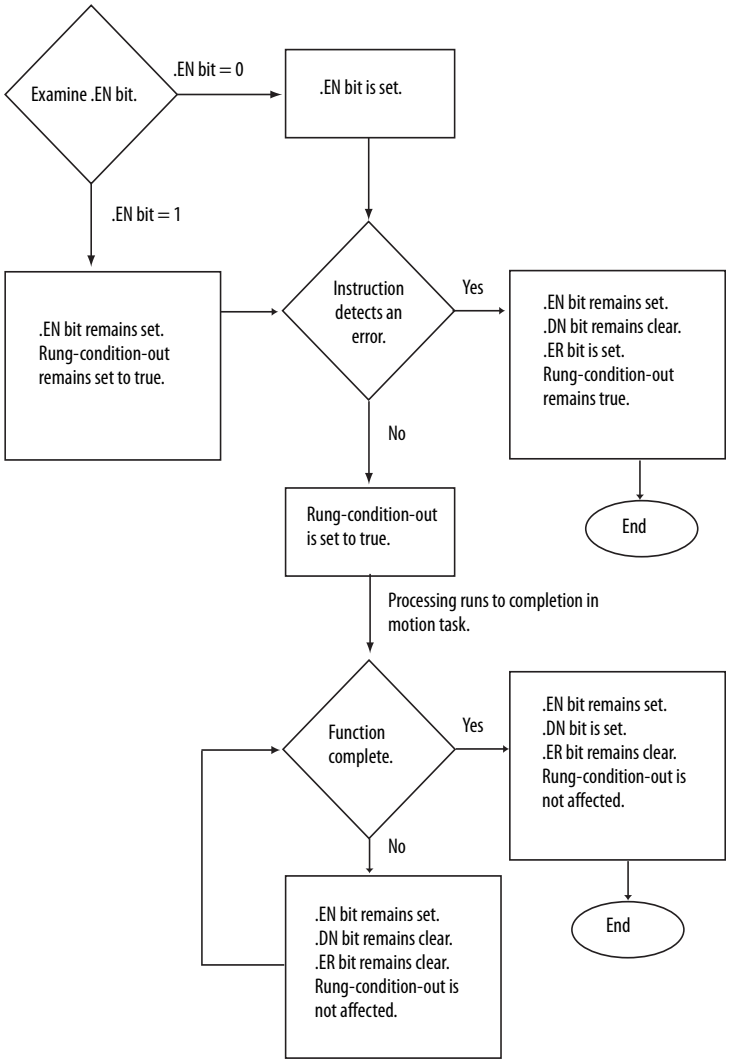
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MDO Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MDO Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MDO instruction receives a Servo Message Failure (12) error message.

Table 20 - MDO Extended Error Codes

Extended Error Code (decimal)	Associated Error Code (decimal)	Meaning
Object Mode conflict (12)	SERVO_MESSAGE_FAILURE (12)	Axis is in shutdown.

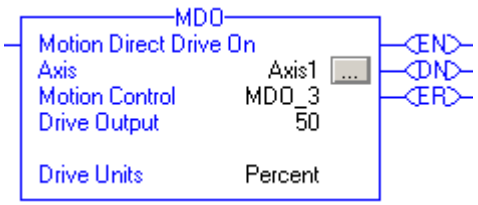
MDO Changes to Status Bits

Bit Name	State	Meaning
DriveEnableStatus	TRUE	Axis is in Drive Control state with the Drive Enable output active.

MDO Example

When the input conditions are true, the controller activates the servo drive for *axis1* and sets the servo output voltage of *axis1*. In this example, the output is 2% of the output value.

Relay Ladder



Structured Text

```
MDO(Axis0,MDO_1,4,percent);
```

Motion Direct Drive Off (MDF)

Use the Motion Direct Drive Off (MDF) instruction to deactivate the servo drive and to set the servo output voltage to the output offset voltage. The output offset voltage is the output voltage that generates zero or minimal drive motion. You can specify this value during axis configuration.

Operands

The MDF instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 21 - MDF Relay Ladder

Operand	Data Type	Description
Axis	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION Tag	Structure used to access instruction status parameters.

Structured Text

MDF(Axis,MotionControl);

The operands are the same as those for the relay ladder MDF instruction.

MOTION_INSTRUCTION Structure

Table 22 - MDF Structured Text

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' drive signals have been successfully disabled and the drive enable status bit is cleared.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

For motion modules with an external servo drive interface, for example the 1756–M02AE, the MDF instruction directly disables the motion module Drive Enable output of the specified physical axis and also "zeroes" the modules' servo output to the external drive by applying the configured Output Offset value.

The MDF instruction is used to stop motion initiated by a preceding Motion Direct Drive On (MDO) instruction and transition the axis from the Direct Drive Control state back to the Axis Ready state.

Executing the Instruction

To successfully execute an MDF instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errors.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

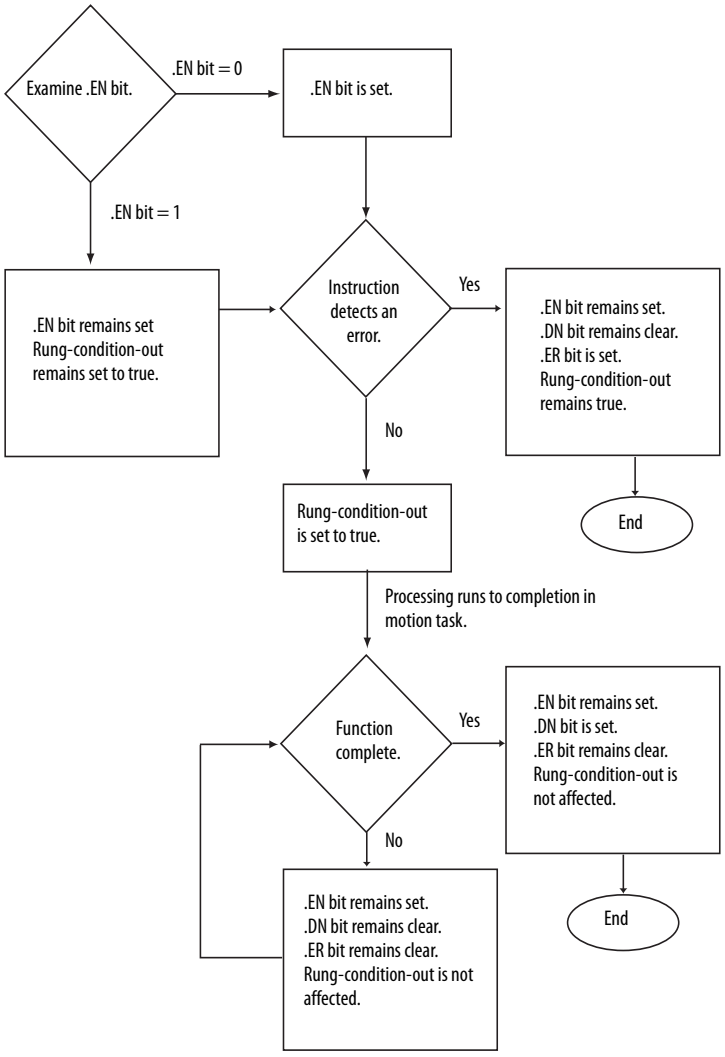
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MDF Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

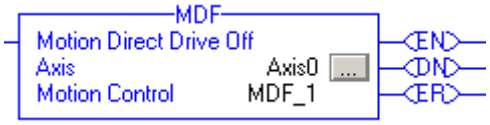
MDF Changes to Status Bits

Bit Name	State	Meaning
DriveEnableStatus	FALSE	Axis is in Axis Ready state with the Drive Enable output now active.

MDF Example

When the input conditions are true, the controller deactivates the servo drive for *axis1* and sets the servo output voltage of *axis_* to the output offset value.

Relay Ladder



Structured Text

```
MDF(Axis0,MDF_1);
```

Motion Drive Start (MDS)

Use the Motion Drive Start (MDS) instruction to activate the drive control loops for the specified CIP axis and run the motor at the specified speed.

IMPORTANT

You can only use the MDS instruction with controllers that support the AXIS_CIP_DRIVE data type.

The MDS instruction is never valid in Position Loop or Feedback Only modes. It is operational in Velocity Loop mode only in drives supporting these ramp attributes(1):

- RampAcceleration
- RampDeceleration
- RampVelocity - Positive
- RampVelocity - Negative
- RampJerk - Control

Torque mode is operational on all CIP drive devices.

(1) Only supported on PowerFlex 700 and 755 drive modules.

Operands

The MDS instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

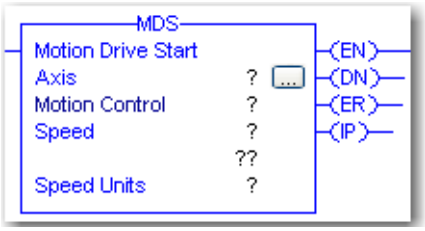


Table 23 - MDS Relay Ladder

Operand	Data Type	Format	Description
Axis	AXIS_CIP_DRIVE	Tag	Motion Axis of data type AXIS_CIP_DRIVE only
Motion control	MOTION_INSTRUCTION	Tag	Structure used to control execution of the motion instruction.
Speed	SINT INT DINT REAL	Immediate or Tag	Defines the initial speed for the DirectVelocityControlStatus Command attribute.
Speed units	SINT INT DINT	Immediate	Which units do you want to use for the speed? 0 = % of Maximum 1 = Units per Sec

Structured Text

MDS(Axis,MotionControl,Speed,Unitspersec);

The operands are the same as those for the relay ladder MDS instruction.

MOTION_INSTRUCTION Structure

Table 24 - MDS MOTION_INSTRUCTION Enumerations Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the rung makes a true-to-false transition and drive messaging is complete.
.DN (Done) Bit 29	It is set when the drive has been successfully enabled and remains set until the rung makes a false-to-true transition.
.ER (Error) Bit 28	It is set when the instruction encounters an error condition. The error condition can be a direct result of the initial conditions or can result during the instruction's execution. The bit remains set until the rung makes a false-to-true transition.
.IP (In process) Bit 26	It is set when the instruction has been successfully initiated and remains set until one of the following occurs: • another MDS instruction supersedes the initial instruction. • another instruction terminates the initial instruction. • a drive fault occurs.
STATE	Reflects the state of the instruction. 0 = Sending a request to the drive module to turn the drive on 1 = Waiting for the drive enable and servo action status bits to be set

Description

The MDS instruction:

- is only valid for the AXIS_CIP_DRIVE axis data type.
- performs a drive enable if the axis is not in the Running state.
- applies desired DirectVelocityControlStatus Command attribute and/or the DirectTorqueControlStatus Command attributes.
- presents the DirectVelocityControlStatus Command attributes and/or the DirectTorqueControlStatus Command attributes.
- is activated on a Rung False-to-True transition.

The MDS instruction is used to activate the direct control of velocity or torque for a specified axis. The instruction performs an axis enable sequence and then presets the DirectVelocityControlStatus Command attribute and/or the DirectTorqueControlStatus Command attribute if the selected drive supports direct control. The most common usage of the MDS instruction is the flying start application, where the following attributes directly control the motion dynamics:

- RampAcceleration
- RampDeceleration
- RampVelocity - Positive
- RampVelocity - Negative
- RampJerk - Control

DirectVelocityControlStatus Command Attribute

The DirectVelocityControlStatus Command attribute is applied by taking the value in the speed field in the instruction configuration and coping it into the DirectVelocityControlStatus Command attribute.

The DirectVelocityControlStatus Command attribute is then scaled and summed onto the commanded output to the drive device. The attribute's value can be modified directly via the MOV instruction.

DirectTorqueControlStatus Command Attribute

The DirectTorqueControlStatus Command attribute is applied by taking the value in speed field in the instruction configuration and coping it into the DirectTorqueControlStatus Command attribute. The DirectTorqueControlStatus Command attribute is then sent directly to the drive via the placeholder in the Controller to Drive Connection.

A combination of the DirectVelocityControlStatus Command attribute and the DirectTorqueControlStatus Command attribute can be applied in applications that require Speed-limited Adjustable Torque (SLAT) modes. SLAT operation mode provides automatic speed control under certain conditions.

The SLAT Configuration is an enumerated attribute that determines how the drive controls torque for this axis instance. To support applications that require SLAT control, the Min/Max torque control enumerations provide a feature to automatically switch to and from speed control under certain conditions. In either SLAT mode the drive operates in one of the states described in this table.

Table 25 - MDS Drive States Descriptions

Enumeration	Mode	Description
0	SLAT disabled	SLAT function is disabled. This is the normal Velocity Loop operation.
1	SLAT Min Speed/Torque	Drive automatically switches from torque control to speed control if Velocity Error > SLAT set point and switches back to torque control if Speed Error < 0.
2	SLAT Max Speed Torque	Drive automatically switches from torque control to speed control if Velocity Error < -SLAT set point and switches back to torque control if Speed Error > 0.

When you execute the MDS instruction and the drive is configured for velocity control, the acceleration and deceleration ramp to the specified speed is controlled by the drives based on the Acceleration Limit and Deceleration Limit attributes. The Motion Planner takes the value from the Direct Command Velocity attribute and sums it into the axis output before sending the command to the drive. The most common use of this instruction is to perform a Drive Start application into a spinning motor, also known as a Flying Start application.

The axis remains in DirectVelocityControlStatus Command attribute or DirectTorqueControlStatus Command attribute modes until cancelled by one of the following instructions:

- Motion Axis Stop (MAS)
- Motion Axis Shutdown (MASD)
- Motion Coordinated Shutdown (MCSD)
- Motion Group Shutdown (MGSD)
- Motion Servo Off (MSF)

Depending on how the fault action is configured, an axis fault can also cancel the MDS instruction.

Execution of the MDS instruction has no effect on motion group or coordinate system objects. However, the instruction affects axis objects as follows:

- When the MDS instruction is initiated without errors, the DirectVelocityControlStatus bit of the MotionStatus axis attribute is set, indicating the DirectVelocityControlStatus bit is active on the axis.

  The DirectVelocityControlStatus bit remains set until it is made inactive via an MAS or MASR instruction, or via an axis fault.

- Also, when the MDS instruction is initiated without errors, the DirectTorqueControlStatus bit attribute of the MotionStatus axis attribute is set, indicating the DirectTorqueControlStatus Command attribute is active on the axis.

  The DirectTorqueControlStatus bit remains set until it is made inactive via an MAS or MASR instruction, or via an axis fault.

Some fault actions impact the execution of the MDS instruction.

Table 26 - MDS Fault Actions Impact Descriptions

Fault Action	Description
Ignore	Ignore instructs the device to completely ignore the exception condition. For some exceptions that are fundamental to the operation of the axis, it can not be possible to Ignore the condition.
Alarm	Alarm action instructs the device to set the associated bit in the Axis Alarm word but to otherwise not affect axis behavior. For some exceptions that are fundamental to the operation of the device, it can not be possible to select this action or any other action that leaves device operation unaffected.
Fault Status Only	Fault Status Only instructs the device to set the associated bit in the Axis Faults word but to otherwise not affect axis behavior. It is up to the controller to programmatically bring the axis to a stop in this condition. For some exceptions that are fundamental to the operation of the device, it can not be possible to select this action or any other action that leaves device operation unaffected.
Stop Planner	Stop Motion instructs the device to set the associated bit in the Axis Faults word and instructs the Motion Planner to perform a controlled stop of all planned motion at the configured Max Decel rate but otherwise not affect axis behavior. This allows the axis to be subsequently moved via the Motion Planner without first clearing the fault. For some exceptions that are fundamental to the operation of the device, it can not be possible to select this action or any other action that leaves device enabled.

Table 26 - MDS Fault Actions Impact Descriptions (Continued)

Fault Action	Description
Stop Drive	The Stop Drive action results in the device both setting the associated bit in the Axis Faults word and bringing the axis to a stop based on the factory set `best' available stopping method. This `best' stopping method includes both the method of decelerating the motor to a stop and the final state of the axis given the expected level of control still available. The level of axis control available depends on the specific exception condition and on the configured control mode. The available deceleration methods are defined by the Stopping Mode attribute. Standard stopping modes, listed in decreasing levels of deceleration control, are as follows: • Ramp Decel • Current Limit Decel • Coast In general, the `best' stopping mode is the most controlled deceleration method still available given the exception condition. The final state of the axis in response to the Major Fault exception action can be any one of the following states that are listed in decreasing levels of control functionality: • Hold (Stopped state with Holding Torque) • Disable (Stopped state with Power Structure Disabled) • Shutdown (DC Bus Power Disabled) The `best' final state of the axis is the state with the most control functionality still available given the exception condition. But in all these final states a fault reset must be executed before the axis can be restored to enabled operation and commanded to move. If the application requires exception action that is a more severe stopping action than the factory set `best' method, the controller must initiate that action. If the application requires exception action that is less severe than the factory set `best' method, the controller must configure the device axis instance for a Minor Fault exception action and handle the fault directly. This can put device and motor components at risk and should only be allowed by the device when there is an opportunity, albeit temporal, for the device to remain operational. This is important in applications where the value of the product is higher than the value of the motor or device. When multiple major faults occur with different stopping actions, the most severe of the associated stopping actions is applied, that is, the stopping action that requires the lowest level of control functionality. This rule also applies to the stopping action associated with the Stopping Mode associated with a Disable Request.
Shutdown	Shutdown forces the axis into the Shutdown state, immediately disabling the drive's power structure. If Shutdown Action is configured to do so, this action also drops DC Bus power to the drive's power structure. Therefore, the Shutdown action overrides the drive's best stopping method. An explicit Shutdown Reset is required to restore the drive to an operational state.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

Error Codes

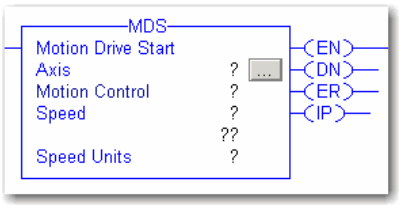
See Error Codes (ERR) for Motion Instructions on page 345.

MDS Changes to Status Bits

Bit Name	State	Meaning
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

MDS Example

Relay Ladder



Structured Text

```
MDS(Axis1,MDS_2,0.25,Unitspersec);
```

Motion Axis Fault Reset (MAFR)

Use the Motion Axis Fault Reset (MAFR) instruction to clear all motion faults for an axis. This is the only method for clearing axis motion faults.

IMPORTANT	The MAFR instruction removes the fault status, but does not perform any other recovery, such as enabling servo action. In addition, when the controller removes the fault status, the condition that generated the fault(s) can still exist. If the condition is not corrected before using the MAFR instruction, the axis immediately faults again.

Operands

The MAFR instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 27 - MAFR Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MAFR(Axis,MotionControl);

The operands are the same as those for the relay ladder MAFR instruction.

MOTION_INSTRUCTION Structure

Table 28 - MAFR Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' faults have been successfully cleared.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MAFR instruction directly clears the specified fault status on the specified axis. It does not correct the condition that caused the error. If the condition is not corrected prior to executing the MAFR instruction the axis could immediately fault again giving the appearance that the fault status was not reset.

This instruction is most commonly used as part of a fault handler program, which provides application specific fault action in response to various potential motion control faults. Once the appropriate fault action is taken, the MAFR instruction can be used to clear all active fault status bits.

To successfully execute a MAFR instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errors.

The MAFR instruction is a procedure type command that is processed from the Logix controller, through the sercos module, and to the associated drives. For applications using a Logix controller with the Logix Designer application, version 13 and earlier, the .DN bit of the instruction was set when the sercos module acknowledged the procedure request. The drives acknowledged the command and processed it quickly. To the user, the .DN bit operation appeared to indicate the successful completion of clearing faults almost immediately.

For applications using a Logix controller with the Logix Designer application, version 15 and later, current drive firmware versions have incorporated a new portion of the fault clearing mechanism that triggers a position recovery process in the drive that recalculates commutation. This recalculation could result in the drive not in a ready state to go servo on for 0.5 to 5 seconds after the reset was performed. To reflect this change in the instruction, the .DN bit operation was changed to wait for the drive to complete the reset procedure. The end result is that you will not see a successful completion of clearing faults almost immediately as you can have in previous versions.

Executing the Instruction

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

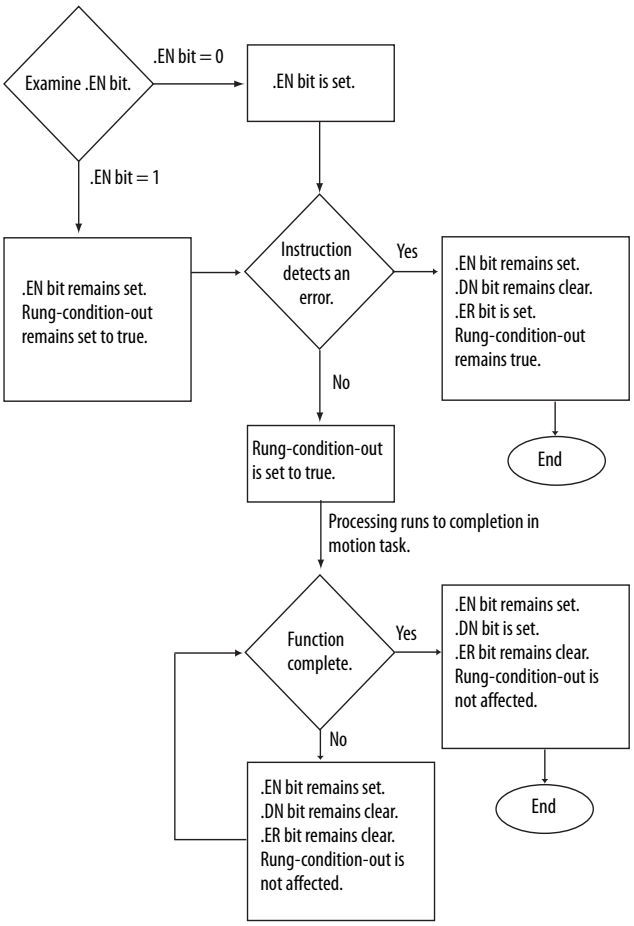
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAFR Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

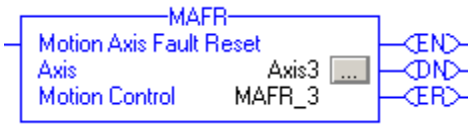
MAFR Changes to Status Bits

The MAFR does not change the status bits.

MAFR Example

When the input conditions are true, the controller clears all motion faults for axis3.

Relay Ladder



Structured Text

```
MAFR(Axis0,MAFR_1);
```

Notes:

Motion Move Instructions

(MAS, MAH, MAJ, MAM, MAG, MCD, MRP, MCCP, MAPC, MATC, MCSV)

ATTENTION: Use tags for the motion control attribute of instructions only once. Reuse of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

Use the Motion Move instructions to control axis position.

Table 29 - Choosing a Motion Move Instruction

If You Want To	Use This Instruction	Page	Languages
Stop any motion process on an axis	Motion Axis Stop (MAS)	82	Relay ladder Structured text
Home an axis	Motion Axis Home (MAH)	92	
Jog an axis	Motion Axis Jog (MAJ)	99	
Move an axis to a specific position	Motion Axis Move (MAM)	111	
Start electronic gearing between two axes	Motion Axis Gear (MAG)	123	
Change the speed, acceleration, or deceleration of a move or a jog that is in progress	Motion Change Dynamics (MCD)	133	
Change the command or actual position of an axis	Motion Redefine Position (MRP)	142	
Calculate a Cam Profile based on an array of cam points	Motion Calculate Cam Profile (MCCP)	150	
Start electronic camming between two axes	Motion Axis Position Cam (MAPC)	156	
Start electronic camming as a function of time	Motion Axis Time Cam (MATC)	178	
Calculate the slave value, slope, and derivative of the slope for a cam profile and master value	Motion Calculate Slave Values (MCSV)	192	

Motion Axis Stop (MAS)

Use the Motion Axis Stop (MAS) instruction to stop a specific motion process on an axis or to stop the axis completely.

Operands

The MAS instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

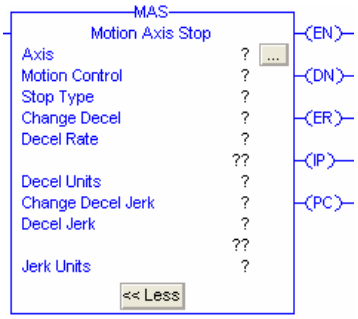


Table 30 - MAS Relay Ladder Operands Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction

Table 30 - MAS Relay Ladder Operands Descriptions (Continued)

Operand	Type	Format	Description	
Stop Type	DINT	Immediate	**To Stop**	**Choose This Stop Type**
			All motion in process for this axis	All = 0 With this selection, the instruction stops all motion on an axis. The stops takes any coordinated motion on the axis into account when it computes the Decel rate and stops that component of the coordinated motion. The other axes components of the coordinated motion are unaffected and continue. If this instruction stops a Motion Drive Start (MDS) instruction, the Direct Control Feature is disabled and the affected axis is decelerated to a stop by using the instruction parameters.(1)
			Only a certain type of motion but leave other motion processes running	Choose the type of motion that you want to stop.
				Jog = 1
				Move = 2
				Gear = 3
				Home = 4
				Tune = 5
				Test = 6
				Position Cam = 8
				Time Cam = 7
				Master Offset Move = 9
				Direct Control = 10(1)
				The axis could still be moving when the MAS instruction is complete.
Change Decel	DINT	Immediate	**If You Want To**	**Then Choose**
			Use the maximum deceleration rate of the axis	No = 0
			Specify the deceleration rate	Yes = 1
Decel Rate	REAL	Immediate Tag	**Important**: The axis could overshoot its target position if you reduce the deceleration while a move is in process. Deceleration rate of the axis. The instruction uses this value only if Change Decel is Yes.	
Decel Units	DINT	Immediate	Which units do you want to use for the Decel Rate? • Units per sec^2 (0) • % of Maximum (1)	
Change Decel Jerk	DINT	Immediate	**If You Want To**	**Then Choose**
			Use the maximum deceleration Jerk rate of the axis	No (0)
			Program the deceleration jerk rate	Yes (1)
Decel Jerk	REAL	Immediate Tag	**Important**: The axis could overshoot its target position if you reduce the deceleration jerk while a move is in process.	
Jerk Units	DINT	Immediate	You **must** enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate of the axis. Use these values to get started. Decel Jerk = 100 (% of Time) 0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started)	

(1) When the MAS instruction is used with either an 'All' or 'DirectVelocityControlStatus Command' Stop Type, the selection also clears the MDS In Process (.IP) bit, and clears the Axis DirectVelocityControlStatus bit in the Motion Status attribute.

Structured Text

MAS(Axis,MotionControl, StopType,ChangeDecel, DecelRate,DecelUnits, ChangeDecelJerk, DecelJerk,JerkUnits);

The structured text operands are the same as the relay ladder operands.

SeeTable 30, MAS Relay Ladder Operands Descriptions on page 82.

For the operands that require you to select from available options, enter your selection as described in Table 31.

Table 31 - MAS Structured Text Operands Descriptions

This Operand	Has These Options, Which You Can	
	Enter as Text	**Or Enter as a Number**
Stop Type	all jog move gear home tune test timecam positioncam masteroffsetmove directcontrol	0 1 2 3 4 5 6 7 8 9 10
Change Decel	no yes	0 1
Decel Units	units per sec^2 % of maximum	0 1
Change Decel Jerk	no yes	0 1
Decel Jerk	no enumeration	You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate of the axis. Use this value to get started: Decel Jerk = 100 % of Time (2)
Jerk Units	unitspersec3 % of maximum % of time	0 1 2

MAS MOTION_INSTRUCTION Structure

Table 32 - MAS Motion_Instruction

Enumerations	Description
.EN (Enable) Bit 31	The EN bit stays set until the process is complete and the rung goes false. A false-to-true transition caused the instruction to execute.
.DN (Done) Bit 29	The stop was successfully initiated.
.ER (Error) Bit 28	An error occurred.
.IP (In Process) Bit 27	The axis is stopping. Any of these actions end the MAS instruction and clear the IP bit: • Axis is stopped • Another MAS instruction supersedes this MAS instruction • Shutdown command • Fault Action
.PC (Process Complete) Bit 26	The axis stopped. The PC bit stays set until the rung makes a false-to-true transition.

Description

Use the MAS instruction when you want a decelerated stop for any controlled motion in process for the axis. The instruction stops the motion without disabling the servo loop. A trapezoidal profile is always used for MAS with Stop Type=ALL for the deceleration regardless of the programmed profile type. Use the instruction to do the following.

- Stop a specific motion process, such as jogging, moving, or gearing.
- Stop the axis completely.
- Abort a test or tune process initiated by a Motion Run Hookup Diagnostics (MRHD) instruction or Motion Run Axis Tuning (MRAT) instruction.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if one of these occurs:

- Maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- Maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

For more information, see Analyzing Axis Motion on page 331.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time you want to execute the instruction.
- In structured text, condition the instruction so that it executes only on a transition by either a qualifyer of an SFC action or structured text construct.

  For more information, see Structured Text Programming on page 359.

Programming Guidelines

Follow these guidelines when programming an MAS instruction.

Stop Motion

Use this instruction to stop a specific type of motion or to stop all motion. The instruction uses a trapezoidal profile and stops the axis.

When you want to stop only a certain type of motion but leave other motion processes running, the axis could still be moving when the MAS instruction is complete. The instruction uses an S-curve profile to stop the axis only if the Stop Type is Jog or Move.

Stop Gearing or Camming

Stop a gearing or position camming process by entering the slave axis to turn off the specific process and stop the axis. If the master axis is a servo axis, you can stop the master axis, which in turn stops the slave without disabling the gearing or position camming.

TIP If the Master axis is moved manually while the Master axis was in a disabled state, the actual position of the slave axis continues to track the Master's position, regardless whether the MasterReference of the MAPC instruction is set to Actual or Command.

Stop a Master Offset Move

Stop an Absolute or Incremental Master Offset move for Axis by entering the slave axis. For deceleration and jerk, enter the values and units for the master axis.

Instruction Changes Motion Parameters

When you execute an MAS instruction, the axis uses the new deceleration and jerk rates for the motion that's already in process. This can cause an axis to overshoot its speed, overshoot its end position, or reverse direction. S-curve profiles are more sensitive to parameter changes.

For more information, see Analyzing Axis Motion on page 331.

Use Jerk Operands

Use the jerk operands when the Stop Type is Jog or Move or the jog or move uses an S-curve profile.

Under those conditions, the instruction uses an S-curve profile to stop the axis. The instruction uses a constant deceleration rate for all other types of stops. You must fill in the jerk operands regardless of the type of stop.

Use % of Time

To program and tune jerk, enter it as a percent of the acceleration or deceleration time.

For more information, see Tune an S-curve Profile on page 327.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAS Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAS Extended Error Codes

Use Extended Error Codes (EXERR) for more information about an error.

Table 33 - MAS Extended Error Codes

If ERR is	And EXERR is	Then		
		Cause	Corrective Action	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 4, then check the Decel Rate.	
			EXERR	**MAS Operand**
			0	Axis
			1	Motion Control
			2	Stop Type
			3	Change Decel
			4	Decel Rate

MAS Changes to Status Bits

If the Stop Type is	Then		
NOT All	The instruction clears the Motion Status bit for the motion process that you stopped.		
All	The instruction clears all Motion Status bits.		
	Bit	**Status**	**Meaning**
	MoveStatus	FALSE	Axis is not moving
	JogStatus	FALSE	Axis is not jogging
	GearingStatus	FALSE	Axis is not gearing
	HomingStatus	FALSE	Axis is not homing
	StoppingStatus	TRUE	Axis is stopping
	PositionCamStatus	FALSE	Axis is not position camming
	TimeCamStatus	FALSE	Axis is not time camming
	PositionCamPendingStatus	FALSE	Axis does not have a position cam pending
	TimeCamPendingStatus	FALSE	Axis does not have a time cam pending
	GearingLockStatus	FALSE	Axis is not in a gear locked condition
	PositionCamLockStatus	FALSE	Axis is not in a cam locked condition
	DirectVelocityControlStatus	FALSE	Axis is not under direct velocity control
	DirectTorqueControlStatus	FALSE	Axis is not under direct torque control

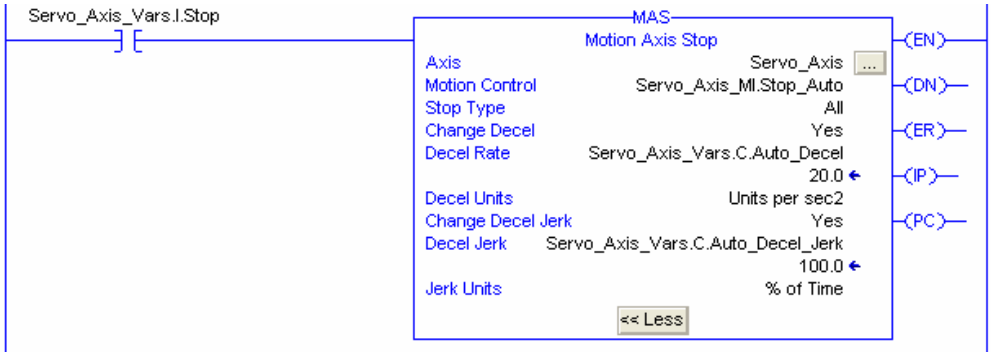
Example 1

Stop All Motion—Relay Ladder

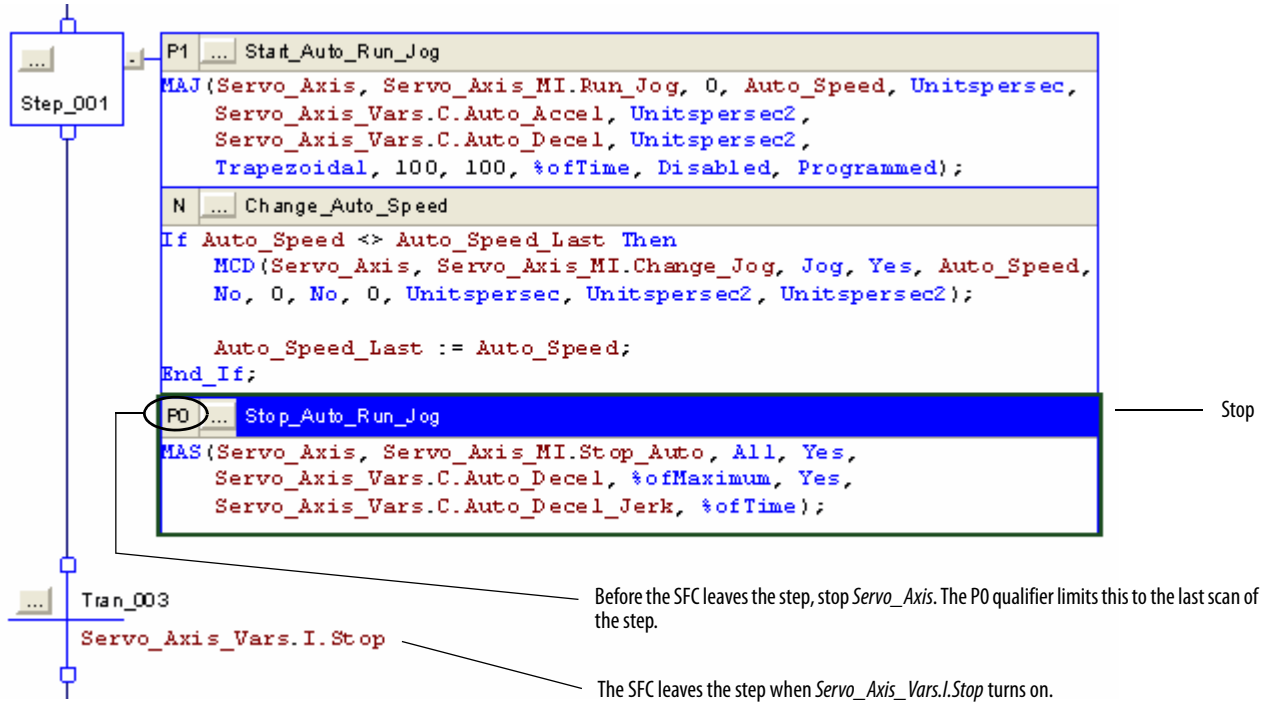
When *Servo_Axis_Vars.I.Stop* turns on, do the following:

- Stop all motion on *Servo_Axis*.
- Decelerate at 20.0 units per sec2.

The instruction doesn't use the Decel Jerk value. Because the Stop Type is all, the instruction uses a trapezoidal profile to stop the axis.



Stop All Motion—Structured Text


Step_001
P1 Start_Auto_Run_Jog
MAJ(Servo_Axis, Servo_Axis_MI.Run_Jog, 0, Auto_Speed, Unitspersec,
Servo_Axis_Vars.C.Auto_Accel, Unitspersec2,
Servo_Axis_Vars.C.Auto_Decel, Unitspersec2,
Trapezoidal, 100, 100, %ofTime, Disabled, Programmed);
N Change_Auto_Speed
If Auto_Speed <> Auto_Speed_Last Then
MCD(Servo_Axis, Servo_Axis_MI.Change_Jog, Jog, Yes, Auto_Speed,
No, 0, No, 0, Unitspersec, Unitspersec2, Unitspersec2);
Auto_Speed_Last := Auto_Speed;
End_If;
P0 Stop_Auto_Run_Jog
MAS(Servo_Axis, Servo_Axis_MI.Stop_Auto, All, Yes,
Servo_Axis_Vars.C.Auto_Decel, %ofMaximum, Yes,
Servo_Axis_Vars.C.Auto_Decel_Jerk, %ofTime);
Stop
Tran_003
Servo_Axis_Vars.I.Stop
Before the SFC leaves the step, stop Servo_Axis. The P0 qualifier limits this to the last scan of the step.
The SFC leaves the step when Servo_Axis_Vars.I.Stop turns on.

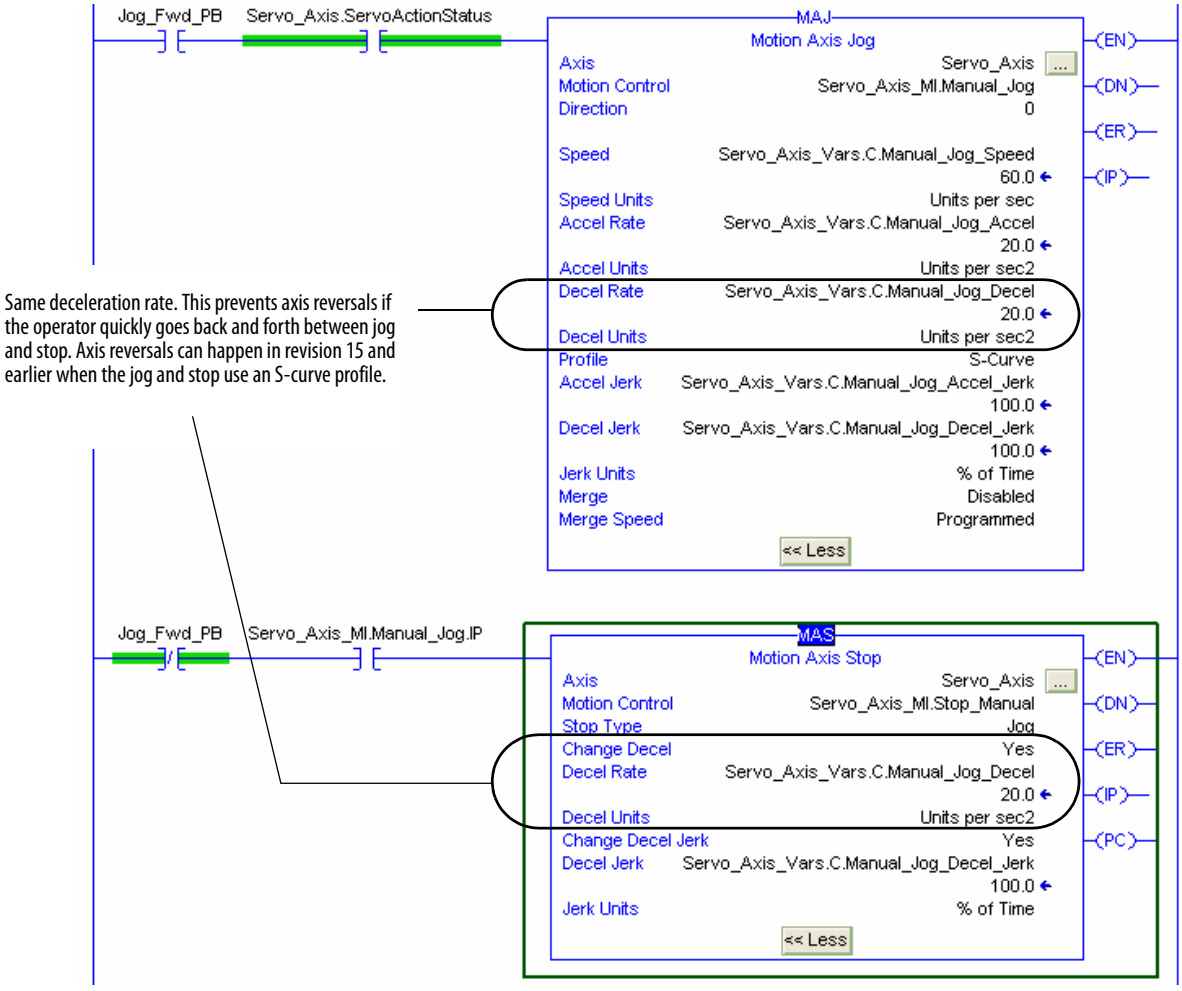
Example 2

Jog Forward with Axis Stop—Relay Ladder

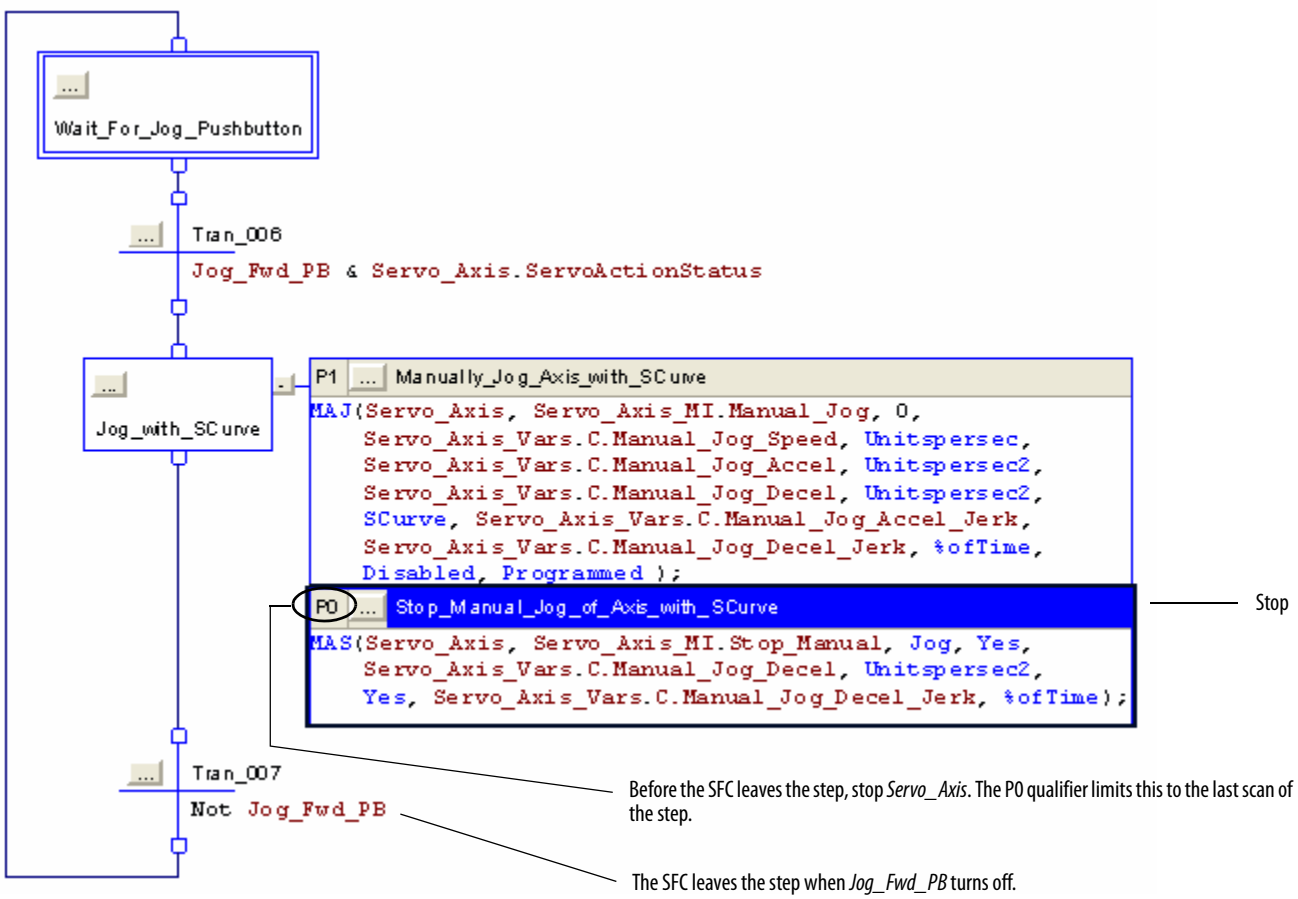
The operator uses a pushbutton to jog an axis. The pushbutton turns the *Jog_Fwd_PB* tag on and off. When the operator releases the button, the MAS instruction stops the axis.

The MAS instruction uses an S-curve profile to stop the axis because of the following:

- The Motion Axis Jog (MAJ) instruction uses an S-curve profile.
- The Stop Type is Jog for the MAS instruction.



Jog Forward with Axis Stop—Structured Text



Motion Axis Home (MAH)

Use the Motion Axis Home (MAH) instruction to home an axis. Two different homing modes can be selected during axis configuration: Active or Passive. If an Active homing sequence is selected, the axis executes the configured Home Sequence Type and establishes an absolute axis position. If Passive homing is selected, however, no specific homing sequence is executed and the axis is left waiting for the next switch, marker, or switch-marker pulse to establish the home position.

Operands

The MAH instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

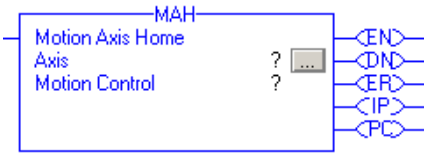


Table 34 - MAH Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MAH(Axis,MotionControl);

The operands are the same as those for the relay ladder MAH instruction.

See Table 34, MAH Relay Ladder Operand Descriptions on page 92.

For the operands that require you to select from available options, enter your selection as described in Table 35.

MAH MOTION_INSTRUCTION Structure

Table 35 - MAH Motion_Instruction Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis home has been successfully completed or is aborted.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 27	It is set on positive rung transition and cleared after the Motion Home Axis is complete, or terminated by a stop command, shutdown, or a servo fault.
.PC (Process Complete) Bit 26	It is set when axis home is successfully completed.

Description

The MAH instruction is used to calibrate the absolute position of the specified axis. For axes that are configured as type Servo, the axis can be homed by using Active, Passive, or Absolute Homing mode configuration. For Feedback Only axes, only Passive and Absolute homing modes are available. Absolute Homing mode requires the axis to be equipped with absolute feedback device.

IMPORTANT	Absolute Homing mode is not available with a CIP axis. However, any successful Home sequence establishes an absolute position.

For CIP axes only, software overtravels are disabled if the Home status bit transitions to the FALSE state.

The MAH instructions sets the HomedStatus bit upon successful completion of the configured homing sequence. This bit indicates that an absolute machine reference position has been established. When this bit is set, operations that require a machine reference, such as Software Overtravel checking can be meaningfully enabled.

For non-CIP Drive axis data types, the HomedStatus bit is cleared under the following conditions:

- Download
- Control power cycle
- Reconnection to Motion Module
- Feedback Loss Fault
- Shutdown

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if one of these occurs:

- Maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- Maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in percent of time.

For more information, see Analyzing Axis Motion on page 331.

Executing the Instruction

To successfully execute an MAH instruction on an axis configured for Active Homing mode, the targeted axis must be configured as a Servo Axis Type. To successfully execute an MAH instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. The MAH instruction also applies to CIP axes configured for Feedback Only, Position Loop, Velocity Loop, or Torque Loop operation. If any of these conditions are not met, the instruction errors.

IMPORTANT The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  See Structured Text Programming on page 359 for more information.

Programming Guidelines

Follow these guidelines when programming an MAH instruction.

Active Homing

When the Axis Homing mode is configured as Active, the physical axis is first activated for servo operation. As part of this process, all other motion in process is cancelled and appropriate status bits are cleared. The axis is then homed by using the configured Home Sequence, which can be Immediate, Switch, Marker, or Switch-Marker. The latter three Home Sequences result in the axis being jogged in the configured Home Direction and then, after the position is redefined based on detection of the home event, the axis is automatically moved to the configured Home Position.

Passive Homing

When the Axis Homing mode is configured as Passive, the MAH instruction redefines the actual position of a physical axis on the next occurrence of the encoder marker. Passive homing is most commonly used to calibrate Feedback Only axes to their markers, switches, or switch-markers, but can also be used on Servo axes. Passive homing is identical to active homing to an encoder marker, except that the motion controller does not command any axis motion.

After initiating passive homing, the axis must be moved past the encoder marker for the homing sequence to complete properly. For closed-loop Servo axes, this can be accomplished with a MAM or MAJ instruction. For physical Feedback Only axes, motion cannot be commanded directly by the motion controller, and must be accomplished via other means.

Absolute Homing

If the motion axis hardware supports an absolute feedback device, Absolute Homing mode can be used. The only valid Home Sequence for an Absolute Homing mode is 'immediate'. In this case, the absolute homing process establishes the true absolute position of the axis by applying the configured Home Position to the reported position of the absolute feedback device. Prior to execution of the absolute homing process via the MAH instruction, the axis must be in the Axis Ready state with the servo loop disabled.

Arithmetic Status Flags

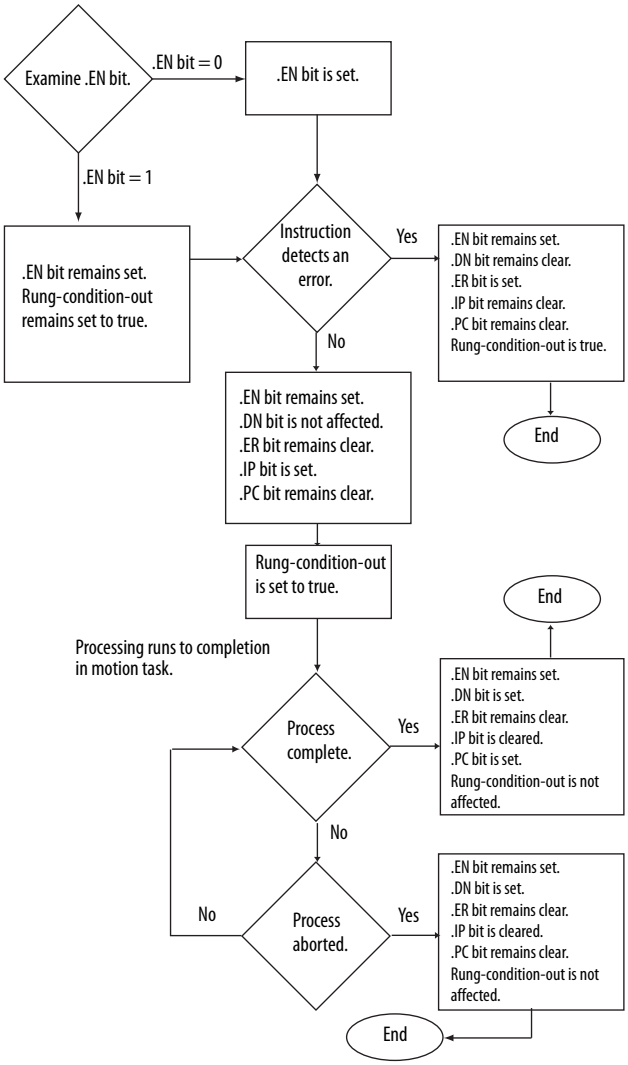
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAH Execution Condition

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



MAH Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAH Extended Error Codes

Use Extended Error Codes (EXERR) for more information about an error. Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions.

The following Extended Error Codes help in troubleshooting when the MAH instruction receives a Servo Message Failure (12) error message or Illegal Homing Configuration (41).

Table 36 - MAH Extended Error Codes Descriptions

If Err is	And EXERR is	Then			Meaning
		Cause	Corrective Action		
12	Varies	Servo Message Failure			
			EXERR	Operand	
			Process Terminated on Request	1	Home execution followed by an instruction to shutdown/disable drive, or a motion stop instruction or a Processor change requests a cancel of Home.
			No Resource	2	Not enough memory resources to complete request. (sercos)
			Object Mode Conflict	12	Axis is in shutdown.
			Permission Denied	15	Enable input switch error. (sercos)
			Device in Wrong State	16	Redefine Position, Home, and Registration 2 are mutually exclusive (sercos), device state not correct for action. (sercos)
41	Varies	Illegal Homing configuration.	Home Sequence	4	The Home Sequence is incompatible with the Home mode. Home speed cannot be zero.
54	-1	The coordinate system has a Maximum Deceleration of 0.	Click the Properties for the coordinate system axis and set a Maximum Deceleration.		The Home Return Speed cannot be zero.
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	If the Extended Error returns a positive number (0-n), it is referring to the offending axis in the coordinate system. Click the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. 1. Click the ellipsis button next to the offending axis to access the Axis Properties screen. 2. Click the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. 3. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Click the Coordinate System Properties Dynamics tab to correct the Maximum Deceleration Value.		

MAH Changes to Status Bits

Bit Name	State	Meaning
HomingStatus	True	Axis is Homing
JogStatus	False	Axis is no longer Jogging
MoveStatus	False	Axis is no longer Moving
GearingStatus	False	Axis is no longer Gearing
StoppingStatus	False	Axis is no longer Stopping

During portions of the active homing sequence, these bits can be set and cleared. The MAH instruction uses the Move and Jog motion profile generators to move the axis during the homing sequence.

This also means that any disruption in the Move or Jog motion profiles due to other motion instructions can affect the successful completion of the MAH initiated homing sequence. If in Passive Homing mode, the MAH instruction sets the Homing Status bit.

Motion Axis Jog (MAJ)

Use the Motion Axis Jog (MAJ) instruction to move an axis at a constant speed until you tell it to stop.

Operands

The MAJ instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

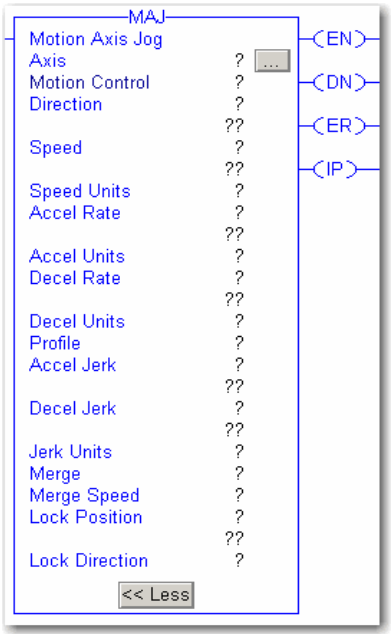


Table 37 - MAJ Relay Ladder Operand Descriptions

Operand	Type	Format	Description	
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to jog.	
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.	
Direction	DINT	Immediate Tag	**For This Jog Direction**	**Enter**
			Forward	0
			Reverse	1
Speed	REAL	Immediate Tag	Speed to move the axis in Speed Units.	
Speed Units	DINT	Immediate	Which units do you want to use for the Speed? 0 = Units per sec 1 = % of Maximum 2= Units per MasterUnit	

Table 37 - MAJ Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Accel Rate	REAL	Immediate Tag	Acceleration rate of the axis in Accel Units.
Accel Units	DINT	Immediate	Which units do you want to use for the Accel Rate? 0 = Units per sec^2 1 = % of Maximum 2= Units per $MasterUnit^2$
Decel Rate	REAL	Immediate Tag	Deceleration rate of the axis in Deceleration Units.
Decel Units	DINT	Immediate	Which units do you want to use for the Decel Rate? 0 = Units per sec^2 1 = % of Maximum 2= Units per $MasterUnit^2$
Profile	DINT	Immediate	Select the velocity profile to run the jog. 0 = Trapezoidal 1 = S-curve
Accel Jerk	REAL	Immediate Tag	You **must** always enter values for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-curve. • Accel Jerk is the acceleration jerk rate for the axis • Decel Jerk is the deceleration jerk rate for the axis. Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time) • Jerk units = 2 Enter the jerk rates in these Jerk Units. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started) 3 = Units per $MasterUnit^3$ 4 = % of Time-Master Driven
Decel Jerk	REAL	Immediate Tag	
Jerk Units	DINT	Immediate	
Merge	DINT	Immediate	Do you want to turn all current axis motion into a pure jog governed by this instruction regardless of the motion instructions currently in process? • NO — Choose Disabled (0) • YES — Choose Enabled (1)
Merge Speed	DINT	Immediate	If Merge is Enabled, which speed do you want to jog at? • Speed of this instruction — Choose Programmed = 0 • Current speed of the axis — Choose Current = 1
Lock Position	REAL or TAG	Immediate	Position on the Master Axis where a Slave axis should start after the move has been initiated on the Slave for Master Driven mode. Not Used for Time Driven mode. 0 = None Master Position Units
Lock Direction	REAL or TAG	Immediate	Specifies when the Master Lock Position should be used for Master Driven mode. Not Used for Time Driven mode. 0 = None 1 = Immediate Forward Only 2 = Immediate Reverse Only 3 = Position Forward Only 4 = Position Reverse Only

Structured Text

MAJ(Axis,MotionControl, Direction,Speed, SpeedUnits,AccelRate, AccelUnits,DecelRate, DecelUnits,Profile, AccelJerk,DecelJerk, JerkUnits,Merge, MergeSpeed, Lock Position, Lock Direction)

The structured text operands are the same as the relay ladder operands.

MAJ Relay Ladder Operand Descriptions on page 99 for more information.

For the operands that require you to select from available options, enter your selection as described in Table 38.

Table 38 - MAJ Structured Text Operand Descriptions

This Operand	Has These Options Which You	
	Enter as Text	**Or Enter as a Number**
Axis	No enumeration	Tag
MotionControl	No enumeration	Tag
Direction	No enumeration	Immediate, Tag
Speed	No enumeration	Immediate, Tag
SpeedUnits	units per sec % of maximum	0 1
AccelRate		
AccelUnits	units per sec^2 % of maximum	0 1
DecelRate		
DecelUnits	units per sec^2 % of maximum	0 1
Profile	trapezoidal S-curve	0 1
AccelJerk	No enumeration	Immediate, Tag You **must** always enter a value for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-curve. Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time)
DecelJerk	No enumeration	
Jerk Units	units per sec^3 % of maximum % of time	0 1 2 (use this value to get started)
Merge	disabled enabled	0 1

Table 38 - MAJ Structured Text Operand Descriptions (Continued)

This Operand	Has These Options Which You	
	Enter as Text	**Or Enter as a Number**
MergeSpeed	programmed current	0 1
Lock Position	Master Lock Position Value	Position on the Master Axis where a Slave axis should start after the move has been initiated on the Slave. Master Units
Lock Direction	Master Lock Position Value	0 = None 1 = Immediate Forward Only 2 = Immediate Reverse Only 3 = Position Forward Only 4 = Position Reverse Only

MAJ MOTION_INSTRUCTION Structure

Table 39 - MAJ Motion_Instruction Enumeration Descriptions

Enumeration	Description
.EN (Enable) Bit 31	A false-to-true transition caused the instruction to execute. The EN bit stays set until the process is complete and the rung goes false.
.DN (Done) Bit 29	The jog was successfully initiated.
.ER (Error) Bit 28	An error occurred.
.IP (In Process) Bit 27	The axis is jogging. Any of these actions stop this jog and clear the IP bit: • Another MAJ instruction supersedes this MAJ instruction • Motion Axis Stop (MAS) instruction • Merge from another instruction • Shutdown command • Fault Action

Description

Use the MAJ instruction to move an axis at a constant speed without regard to position.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if one of these occurs:

- Maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- Maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

For more information, see Analyzing Axis Motion on page 331.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung condition each time you want to execute the instruction.
- In structured text, instructions execute each time they are scanned.

See Structured Text Programming on page 359 for more information.

Condition the instruction so that it only executes on a transition. Use either of these methods:

- Qualifier of an SFC action
- Structured text construct

Programming Guidelines

Follow these guidelines when programming an MAJ instruction.

TIP If the Master axis is moved manually while the Master axis was in a disabled state, the actual position of the slave axis continues to track the Master's position, regardless whether the MasterReference of the MAPC instruction is set to Actual or Command.

Jerk Operands

Use the jerk operands when the instruction uses an S-curve profile. You must fill in the jerk operands regardless of the profile.

% of Time

Use % of Time to program and tune jerk, enter it as a % of the acceleration or deceleration time.

For more information, see Tune an S-curve Profile on page 327.

Cancel Motion with Merge

If you want to add the jog to any motion already in process, use the following:

- Merge = Disabled
- Merge Speed = Programmed

The instruction ignores Merge Speed but you must fill it in anyway.

End Motion from other Instructions

If you want to end the motion from other instructions and you want to just jog at the speed that you set in this instruction, use the following.

- Merge = Enabled
- Merge Speed = Programmed

If you want to end the motion from other instructions and jog at the speed that the axis is already moving at, use the following.

- Merge = Enabled
- Merge Speed = Current

The Speed and Direction operands on the MAJ faceplate are ignored for MAJ for Merge current. Direction parameter is ignored for a merge current.

Starting another Jog

If you start a new MAJ instruction while one is already in process, in revision 15 and earlier you can cause the following.

- An accelerating axis to overshoot its speed
- A decelerating axis to reverse

This happens if the MAJ instructions use an S-curve profile. The reason for this is that the new MAJ instruction cancels the old MAJ instruction. The axis uses the speed, acceleration, deceleration, and jerk of the new instruction.

For more information, see Analyzing Axis Motion on page 331.

Stop a Jog

Use an MAS instruction to stop the jog. See the examples on page 89.

Change the Speed

Use an MCD instruction to change the speed while jogging. See the examples on page 141.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAJ Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAJ Extended Error Codes

Use Extended Error Codes (EXERR) for more information about an error.

Table 40 - MAJ Extended Error Codes Description

If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed
54	-1	The coordinate system has a Maximum Deceleration of 0.	Click the Properties for the coordinate system axis and set a Maximum Deceleration.	
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	1. Open the Properties for the axis. 2. Use the EXERR value to see which axis has the Maximum Deceleration of 0. 3. The axis that you are jogging has a deceleration rate of 0.	

MAJ Changes to Status Bits

If Merge is	Then the Instruction Changes These Bits		
	Bit Name	**State**	**Meaning**
Disabled	JogStatus	TRUE	The axis is Jogging.
Enabled	JogStatus	TRUE	The axis is Jogging.
	MoveStatus	FALSE	The axis is no longer Moving.
	GearingStatus	FALSE	The axis is no longer Gearing.

Example 1

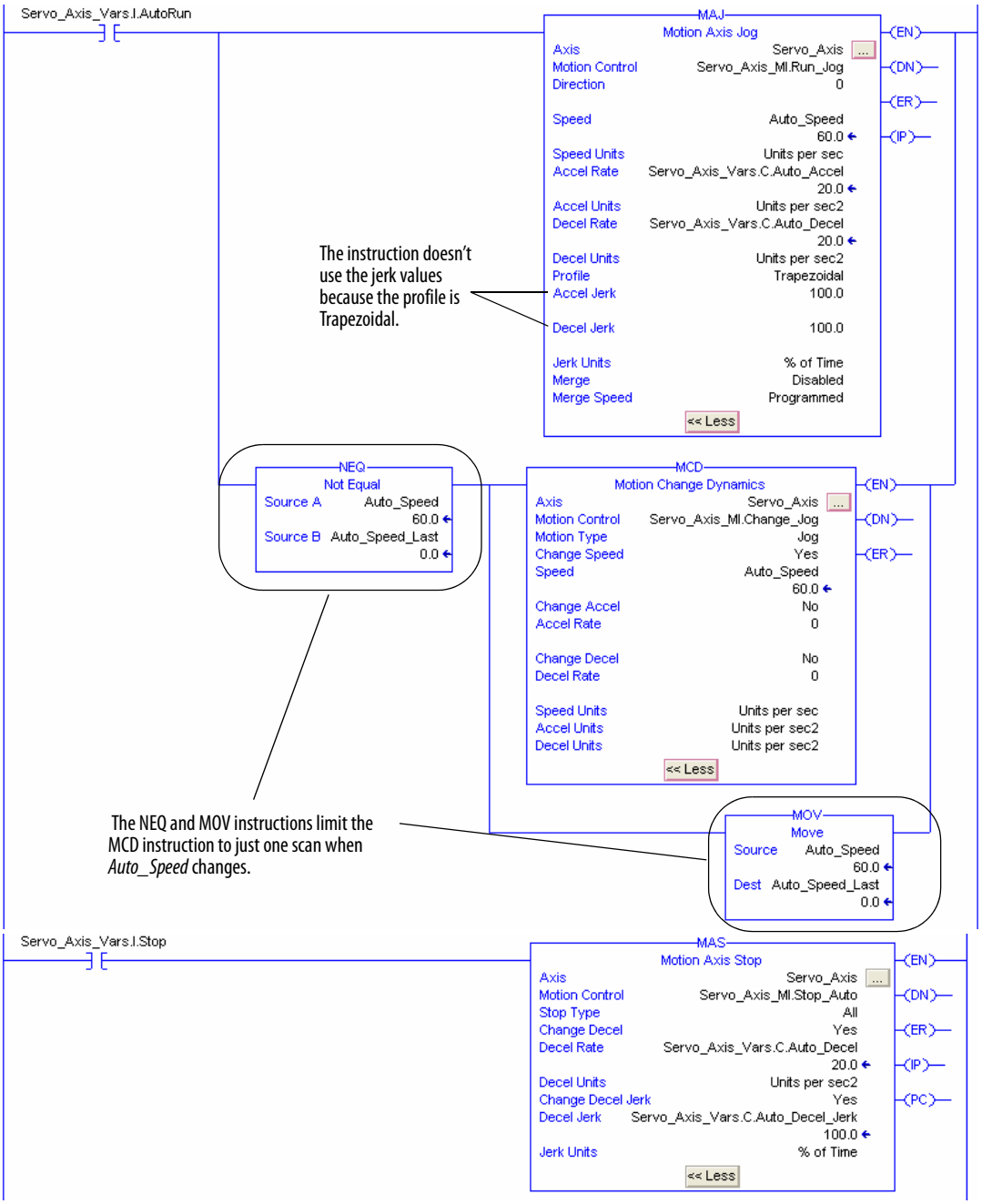
Jog with Speed Change—Relay Ladder

When *Servo_Axis_Vars.I.AutoRun* turns on

Run *Servo_Axis* at *Auto_Speed*.

If *Auto_Speed* changes then change the speed of the jog to the new value of *Auto_Speed*.

When *Servo_Axis_Vars.I.Stop* turns on, stop *Servo_Axis*.



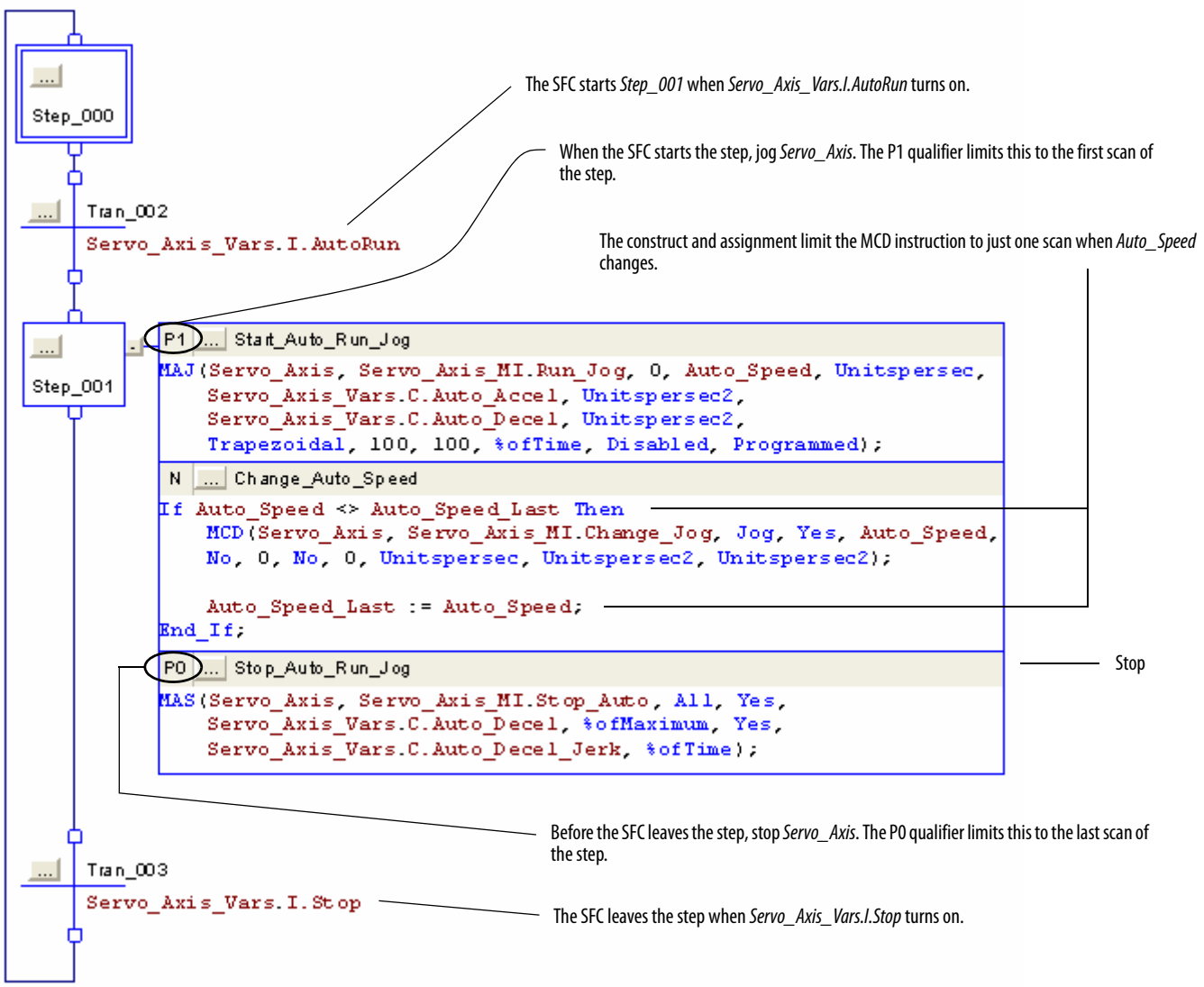
Jog with Speed Change—Structured Text

When *Servo_Axis_Vars.I.AutoRun* turns on

Run *Servo_Axis* at *Auto_Speed*.

If *Auto_Speed* changes then change the speed of the jog to the new value of *Auto_Speed*.

When *Servo_Axis_Vars.I.Stop* turns on, stop *Servo_Axis*.


Step_000
Tran_002
Servo_Axis_Vars.I.AutoRun
Step_001
P1
Start_Auto_Run_Jog
MAJ(Servo_Axis, Servo_Axis_MI.Run_Jog, 0, Auto_Speed, Unitspersec,
Servo_Axis_Vars.C.Auto_Accel, Unitspersec2,
Servo_Axis_Vars.C.Auto_Decel, Unitspersec2,
Trapezoidal, 100, 100, %ofTime, Disabled, Programmed);
N
Change_Auto_Speed
If Auto_Speed <> Auto_Speed_Last Then
MCD(Servo_Axis, Servo_Axis_MI.Change_Jog, Jog, Yes, Auto_Speed,
No, 0, No, 0, Unitspersec, Unitspersec2, Unitspersec2);
Auto_Speed_Last := Auto_Speed;
End_If;
P0
Stop_Auto_Run_Jog
MAS(Servo_Axis, Servo_Axis_MI.Stop_Auto, All, Yes,
Servo_Axis_Vars.C.Auto_Decel, %ofMaximum, Yes,
Servo_Axis_Vars.C.Auto_Decel_Jerk, %ofTime);
Tran_003
Servo_Axis_Vars.I.Stop
The SFC starts Step_001 when Servo_Axis_Vars.I.AutoRun turns on.
When the SFC starts the step, jog Servo_Axis. The P1 qualifier limits this to the first scan of the step.
The construct and assignment limit the MCD instruction to just one scan when Auto_Speed changes.
Stop
Before the SFC leaves the step, stop Servo_Axis. The P0 qualifier limits this to the last scan of the step.
The SFC leaves the step when Servo_Axis_Vars.I.Stop turns on.

Example 2

Jog Forward and Reverse with S-curve—Relay Ladder

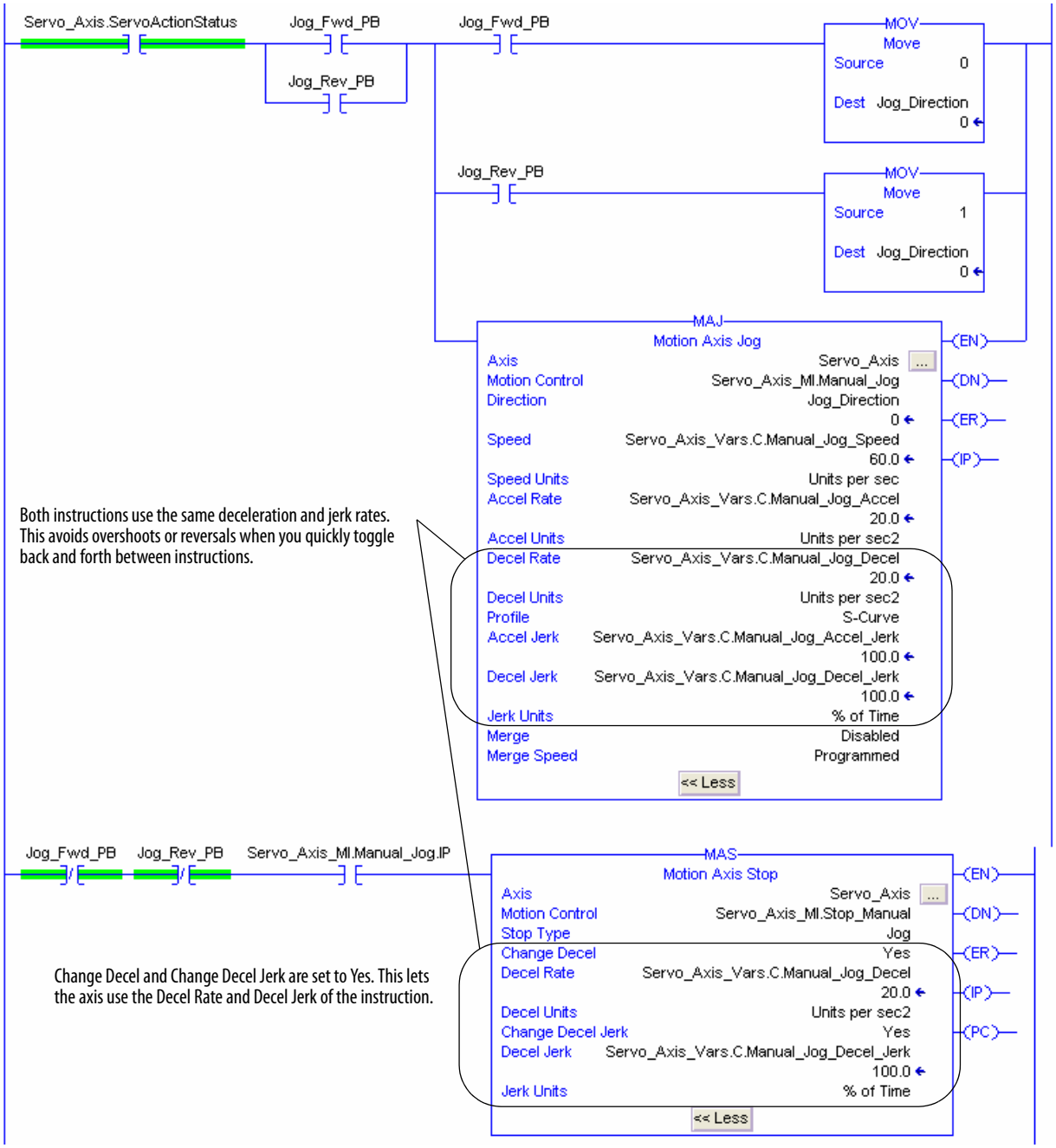
When the servo loop is enabled

And *Jog_Fwd_PB* or *Jog_Rev_PB* turn on

Set *Jog_Direction*.

Run *Servo_Axis* at *Servo_Axis_Vars.C.Manual_Jog_Speed*.

When *Jog_Fwd_PB* and *Jog_Rev_PB* are off, stop *Servo_Axis*.



Jog Forward and Reverse with S-curve—Structured Text

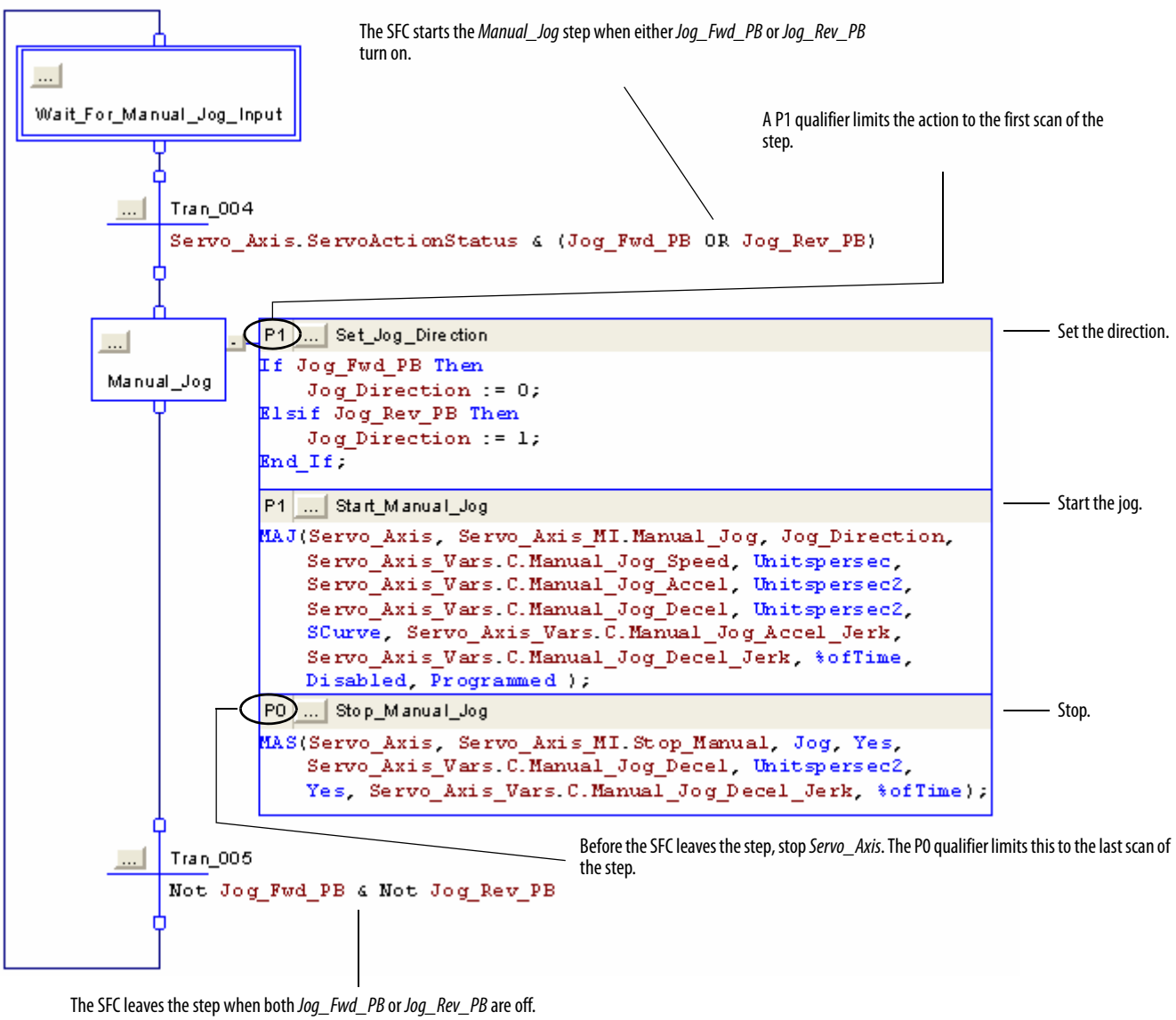
When the servo loop is enabled

And *Jog_Fwd_PB* or *Jog_Rev_PB* turn on

Set *Jog_Direction*.

Run *Servo_Axis* at *Servo_Axis_Vars.C.Manual_Jog_Speed*.

When *Jog_Fwd_PB* and *Jog_Rev_PB* are off, stop *Servo_Axis*.



Motion Axis Move (MAM)

Use the Motion Axis Move (MAM) instruction to move an axis to a specified position.

Operands

The MAM instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

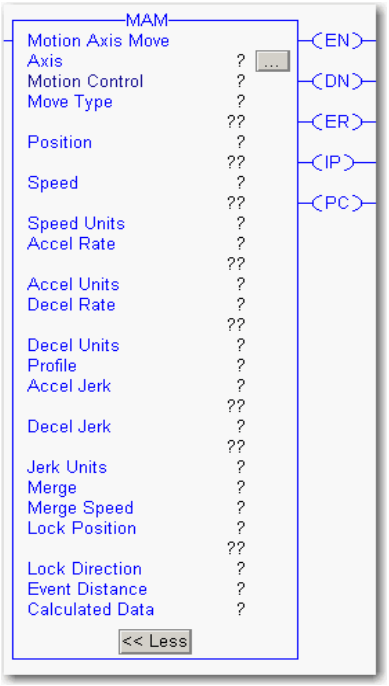


Table 41 - MAM Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis. For an Absolute or Incremental Master Offset move, enter the slave axis.
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.

Table 41 - MAM Relay Ladder Descriptions (Continued)

<table>
<tr><th>Operand</th><th>Type</th><th>Format</th><th colspan="3">Description</th></tr>
<tr><td rowspan="9">Move Type</td><td rowspan="9">DINT</td><td rowspan="9">Immediate
Tag</td><td>To</td><td>Use This Move Type</td><td>And enter</td></tr>
<tr><td>Move an axis to an absolute position</td><td>Absolute</td><td>0</td></tr>
<tr><td>Move an axis a specified distance from where it is now</td><td>Incremental</td><td>1</td></tr>
<tr><td>Move a Rotary axis to an absolute position in the shortest direction regardless of its current position</td><td>Rotary Shortest Path</td><td>2</td></tr>
<tr><td>Move a Rotary axis to an absolute position in the positive direction regardless of its current position</td><td>Rotary Positive</td><td>3</td></tr>
<tr><td>Move a Rotary axis to an absolute position in the negative direction regardless of its current position</td><td>Rotary Negative</td><td>4</td></tr>
<tr><td>Off set the master value of a position cam to an absolute position</td><td>Absolute Master Offset</td><td>5</td></tr>
<tr><td>Off set the master value of a position cam by an incremental distance</td><td>Incremental Master Offset</td><td>6</td></tr>
<tr><td colspan="3">See Choose a Move Type for a Rotary Axis on page 117 for more information about rotary moves.</td></tr>
<tr><td rowspan="9">Position</td><td rowspan="9">REAL</td><td rowspan="9">Immediate
Tag</td><td colspan="3">Absolute position or incremental distance for the move</td></tr>
<tr><td>For this Move Type</td><td colspan="2">Enter this Position value</td></tr>
<tr><td>Absolute</td><td colspan="2">Position to move to</td></tr>
<tr><td>Incremental</td><td colspan="2">Distance to move</td></tr>
<tr><td>Rotary Shortest Path</td><td colspan="2" rowspan="3">Position to move to. Enter a positive value that is less than the Position Unwind value.</td></tr>
<tr><td>Rotary Positive</td></tr>
<tr><td>Rotary Negative</td></tr>
<tr><td>Absolute Master Offset</td><td colspan="2">Absolute offset position</td></tr>
<tr><td>Incremental Master Offset</td><td colspan="2">Incremental offset distance</td></tr>
<tr><td>Speed</td><td>REAL</td><td>Immediate
Tag</td><td colspan="3">Speed to move the axis in Speed Units.
0 = Units per sec
1 = % of Maximum
2 = Seconds
3 = Units per MasterUnit
4 = Master Units</td></tr>
<tr><td>Speed Units</td><td>DINT</td><td>Immediate</td><td colspan="3">Which units do you want to use for the Speed?
0 = Units per sec
1 = % of Maximum</td></tr>
<tr><td>Accel Rate</td><td>REAL</td><td>Immediate
Tag</td><td colspan="3">Acceleration rate of the axis in Accel Units</td></tr>
<tr><td>Accel Units</td><td>DINT</td><td>Immediate</td><td colspan="3">Which units do you want to use for the Accel Rate?
0 = Units per sec^2
1 = % of Maximum
2 = Seconds
3 = Units per MasterUnit^2
4 = Master Units</td></tr>
</table>

Table 41 - MAM Relay Ladder Descriptions (Continued)

Operand	Type	Format	Description
Decel Rate	REAL	Immediate Tag	Deceleration rate of the axis in Deceleration Units.
Decel Units	DINT	Immediate	Which units do you want to use for the Decel Rate? 0 = Units per sec^2 1 = % of Maximum 2 = Seconds 3 = Units per $MasterUnit^2$ 4 = Master Units
Profile	DINT	Immediate	Select the velocity profile to run for the move. 0 = Trapezoidal 1 = S-curve
Accel Jerk	REAL	Immediate Tag	The instruction only uses the jerk operands if the Profile is S-curve. You must always fill them in however. • Accel Jerk is the acceleration jerk rate for the axis. • Decel Jerk is the deceleration jerk rate for the axis. Use these values to get started. • Accel Jerk = 100 • Decel Jerk = 100 • Jerk Units = 2 (% of Time) You can also enter the jerk rates in these Jerk Units. • Units per sec^3 (0) • % of Maximum (1) When the MAM instruction is configured for S-curve operation, Jerk parameters have a 1% of Time minimum value. Setting a MAM to S-curve with 0% of Time Jerk values will not produce a Trapezoidal profile.
Decel Jerk	REAL	Immediate Tag	
Jerk Units	DINT	Immediate	
Merge	DINT	Immediate	Do you want to turn all current axis motion into a pure move governed by this instruction regardless of the motion instructions currently in process? • NO — Choose Disabled (0) • YES — Choose Enabled (1)
Merge Speed	DINT	Immediate	If Merge is Enabled, which speed do you want to move at? • Speed of this instruction — Choose Programmed (0) • Current speed of the axis — Choose Current (1)
Lock Position	REAL	Immediate	Position on the Master Axis where a Slave axis should start after the move has been initiated on the Slave. Master Units
Lock Direction	DINT	Immediate	Specifies when the Master Lock Position should be used. 0 = None 1 = Immediate Forward Only 2 = Immediate Reverse Only 3 = Position Forward Only 4 = Position Reverse Only
Event Distance	REAL	Immediate	Tag array, where Event Distance is specified. 0 = Default, not used or 0. 0 is not an array but a constant for both event distance or calculated data.
Calculated Data	REAL	Immediate	Tag array, where event Calculated Data is specified. Default = 0

Structured Text

MAM(Axis, MotionControl, MoveType, Position, Speed, SpeedUnits, AccelRate, AccelUnits, DecelRate, DecelUnits, Profile, AccelJerk, DecelJerk, JerkUnits, Merge, MergeSpeed, LockPosition, LockDirection, EventDistance, CalculatedData)

The operands are the same as the relay ladder instruction.

See Table 41, MAM Relay Ladder Descriptions on page 111.

MAM MOTION_INSTRUCTION Structure

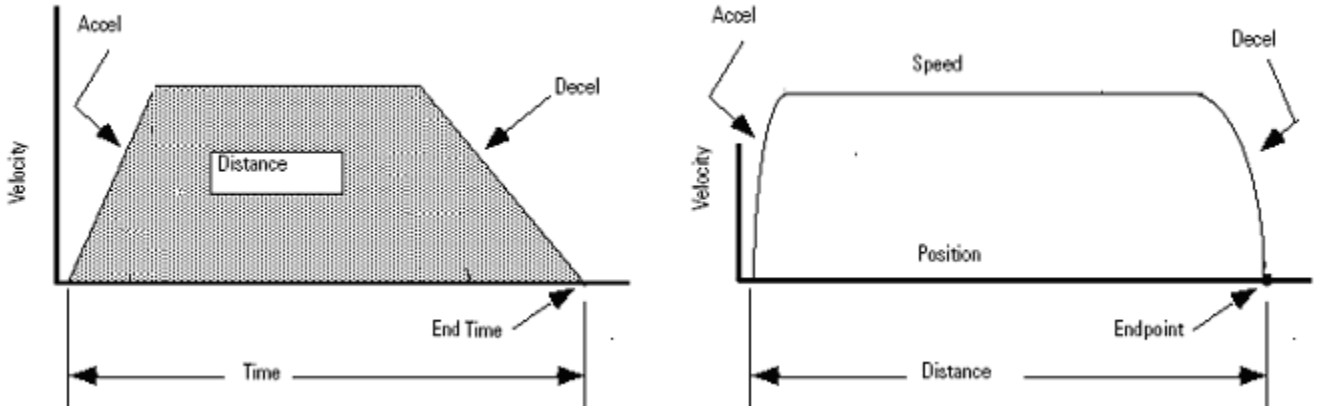
Table 42 - MAM Motion_Instruction Enumeration Descriptions

Enumeration	Description
.EN (Enable) Bit 31	A false-to-true transition caused the instruction to execute. The EN bit stays set until the process is complete and the rung goes false.
.DN (Done) Bit 29	The move was successfully initiated.
.ER (Error) Bit 28	An error occurred.
.IP (In Process) Bit 27	The axis is moving. Any of these actions stop this move and clear the IP bit: • The axis gets to the end Position • Another MAM instruction supersedes this MAM instruction • MAS instruction • Merge from another instruction • Shutdown command • Fault Action
.PC (Process Complete) Bit 27	• The PC bit stays set until the rung makes a false-to-true transition. • The PC bit stays cleared if some other action stops the move before the axis gets to the end Position.

Description

The MAM instruction moves an axis to either a specified absolute position or by a specified incremental distance. The MAM instruction can also produce other special types of moves.

Figure 4 - Trapezoidal Move Starting from Standstill



ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if one of these occurs:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

For more information, see Analyzing Axis Motion on page 331.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung condition each time you want to execute the instruction.
- In structured text, instructions execute each time they are scanned.

  Condition the instruction so that it only executes on a transition. Use either of these methods:

  – Qualifier of an SFC action
  – Structured text construct

  See Structured Text Programming on page 359 for more information.

Programming Guidelines

Follow these guidelines when programming an MAM instruction.

Master Offset Move

For a Master Offset move, enter the slave axis but use master units. Use an Absolute or Incremental Master Offset move to off set the master value of a position cam without actually changing the position of the master axis. This shifts the position cam profile along the master axis.

- For Axis, enter the slave axis.
- For Position, enter the absolute offset position or incremental offset distance
- For Speed, Acceleration, Deceleration, and Jerk, enter them for the master axis.

While the PositionCamLockStatus = 0, Master Offset changes are applied to the master side of the position cam. There will be no corresponding change to the slave axis side of the position cam. The instruction adds in the offset at the Speed, Acceleration, Deceleration, and Jerk values.

% of Time

Use % of Time to program and tune a jerk, enter it as a % of the acceleration or deceleration time.

For more information, see Tune an S-curve Profile on page 327.

Merge to Cancel

These are programming guidelines when you want to use Merge to cancel the motion that is already in process.

- If you want to add the move to any motion already in process, set:
  - Merge = Disabled
  - Merge Speed = Programmed

  The instruction ignores Merge Speed but you must fill it in anyway.
- If you want to end the motion from other instructions and just move at the speed that you set in this instruction, set:
  - Merge = Enabled
  - Merge Speed = Programmed.
- If you want to end the motion from another instruction and just move at the speed that the axis is already moving at, set:
  - Merge = Enabled
  - Merge Speed = Current

  The instruction ignores the value that you put in the Speed operand.

If it is an Absolute or Incremental Master Offset Move, and Merge is Enabled, the move **only** ends an Absolute or Incremental Master Offset move that's already in process. The move **doesn't** affect any other motion that's already in process.

Use a Second MAM Instruction

You can use a second MAM instruction to change one that is already in process. You can change the position, speed, acceleration, or deceleration. The change immediately takes effect.

To change the position of an Absolute Move, set up a second MAM instruction to do either of the following.

- Set the Move Type to Absolute and the Position to the new position.
- Set the Move Type to Incremental and set the Position to the distance to change the end position. The new end position is the old end position plus the new incremental distance.

In either case, the axis moves to the new position without stopping at the old position—including any required change of direction.

Combining a Move

You can combine a move with gearing for complex profiles and synchronization. You can use a Motion Axis Gear (MAG) instruction together with an MAM instruction. This superimposes the gearing on top of the move or the move on top of the gearing. For example, superimpose an incremental move on top of electronic gearing for phase advance and retard control.

Choose a Move Type for a Rotary Axis

Table 43 - Move Types for a Rotary Axis

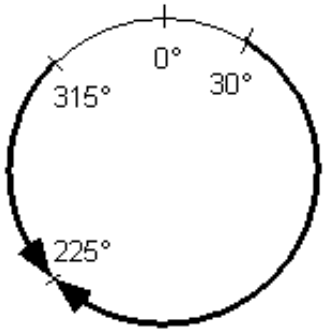
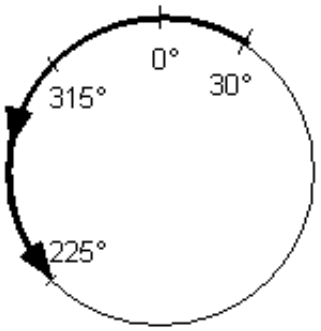
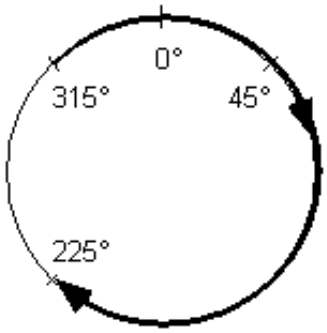
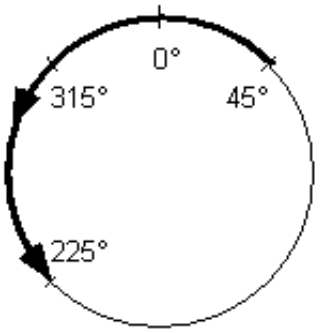
Move Type	Example	Description
Absolute	Absolute move to 225°. The direction depends on the starting position of the axis. 	With an Absolute move, the direction of travel depends on the current position of the axis and isn't necessarily the shortest path to the end position. Starting positions less than the end position result in motion in the positive direction, while starting positions greater than the end position result in motion in the negative direction. The specified position is interpreted trigonometrically and can be positive or negative. It can also be greater than the Position Unwind value. Negative position values are equivalent to their corresponding positive values and are useful when rotating the axis through 0. For example, -90° is the same as +270°. When the position is greater than or equal to the Position Unwind value, the axis moves through more than one revolution before stopping at an absolute position.
Incremental		The specified distance is interpreted trigonometrically and can be positive or negative. It can also be greater than the Position Unwind value. When the distance is greater than the Position Unwind value, the axis moves through more than one revolution before stopping.

Table 43 - Move Types for a Rotary Axis (Continued)

Move Type	Example	Description
Rotary Shortest Path	Rotary Shortest Path move from 30° to 225°. 	**Important**: Only use a Rotary Shortest Path move if the Positioning mode of the axis is Rotary (Rotary axis). A Rotary Shortest Path move is a special type of absolute move for a Rotary axes. The axis: • moves to the specified Position in the **shortest** direction regardless of its current position. • moves through 0° if needed. With a Rotary Shortest Path move, you: • can start the move while the axis is moving or standing still. • **cannot** move the axis more than one revolution with a single move.
Rotary Positive	Rotary Positive move from 315° to 225°. 	**Important**: Only use a Rotary Positive move while the axis is standing still and not moving. Otherwise the axis could move in the wrong direction. A Rotary Positive move is a special type of absolute move for a Rotary axes. The axis: • moves to the specified Position in the **positive** direction regardless of its current position. • moves through 0° if needed. You cannot move the axis more than one revolution with a single Rotary Shortest Path move.
Rotary Negative	Rotary Negative move from 45° to 225°. 	**Important**: Only use a Rotary Negative move while the axis is standing still and not moving. Otherwise the axis could move in the wrong direction. A Rotary Negative move is a special type of absolute move for a Rotary axes. The axis: • moves to the specified Position in the **negative** direction regardless of its current position. • moves through 0° if needed. You cannot move the axis more than one revolution with a single Rotary Shortest Path move.

Pending Move

If there is a pending move on an axis because the master axis has not started, any attempts to move the same axis with a MAM instruction, will take into account the increment from the pending move. For example, if you are currently in MDAC mode. Then you start an incremental move of 10 units on the slave axis, the MAM instruction move will go IP (In Progress bit is set) but no motion takes place because the master has not moved yet.

To start the second move, execute an incremental MAM instruction of 1 unit from motion direct command. The axis will move 11 units. You might expect the axis to move only 1 unit. These steps explain what happens.

1. When the slave axis is waiting for the master to move, the instruction goes IP but no move takes place.
2. The AxisMoveLock Status is enabled.
3. Start the motion direct move.

   AxisLockStatus immediately goes to zero and the move takes place. If the Axis Lock Status goes to zero, it seems that the qued move should be ignored.

This is an example.

1. Start MDAC in Rung 1.
2. Start first move in Rung 2.

   No motion takes place.
3. From Motion Direct command, do an incremental MAM of 1.

   This results in a move of 11 units

Merging in Incremental Mode

The Merge for coordinated motion operates differently from a merge on a Motion Coordinated Linear Move (MCLM) instruction. For the MAM, any uncompleted motion at the point of the merge remains in the move.

For example, assume that you have a single axis MAM programmed in incremental mode from a starting absolute position = 0 and with the programmed incremental distance = 4 units:

- If a merge occurs at an absolute position of 1 and the merge is another incremental move of 4 units, the move completes at a position = 8.
- If this example occurs on a Motion Coordinated Linear Move (MCLM) instruction programmed in incremental mode, the final position = 5.

For more information on how this merge occurs on a MAM programmed in incremental mode, see page 120.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

Extended Error Codes

Use Extended Error Codes (EXERR) for more information about an error.

Table 44 - MAM Extended Errors Codes

If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 4, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Move Type
			3	Position
			4	Speed
54	-1	The coordinate system has a Maximum Deceleration of 0.	Click the Properties for the coordinate system and set a Maximum Deceleration.	
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	1. Open the Properties for the axis. 2. Use the EXERR value to see which axis has the Maximum Deceleration of 0. 3. The axis that you are moving via the MAM instruction has a decel rate of 0.	

MAM Changes to Motion Status Bits

If the Move Type is	And Merge is	Then the Instruction Changes These Bits		
		Bit Name	**State**	**Meaning**
NOT Absolute Master Offset or Incremental Master Offset	Disabled	MoveStatus	TRUE	Axis is Moving
	Enabled	MoveStatus	TRUE	Axis is Moving
		JogStatus	FALSE	Axis is no longer Jogging
		GearingStatus	FALSE	Axis is no longer Gearing
Absolute Master Offset or Incremental Master Offset →		MasterOffsetMoveStatus	TRUE	Axis is Offset
		JogStatus	FALSE	Axis is no longer Jogging
		GearingStatus	FALSE	Axis is no longer Gearing

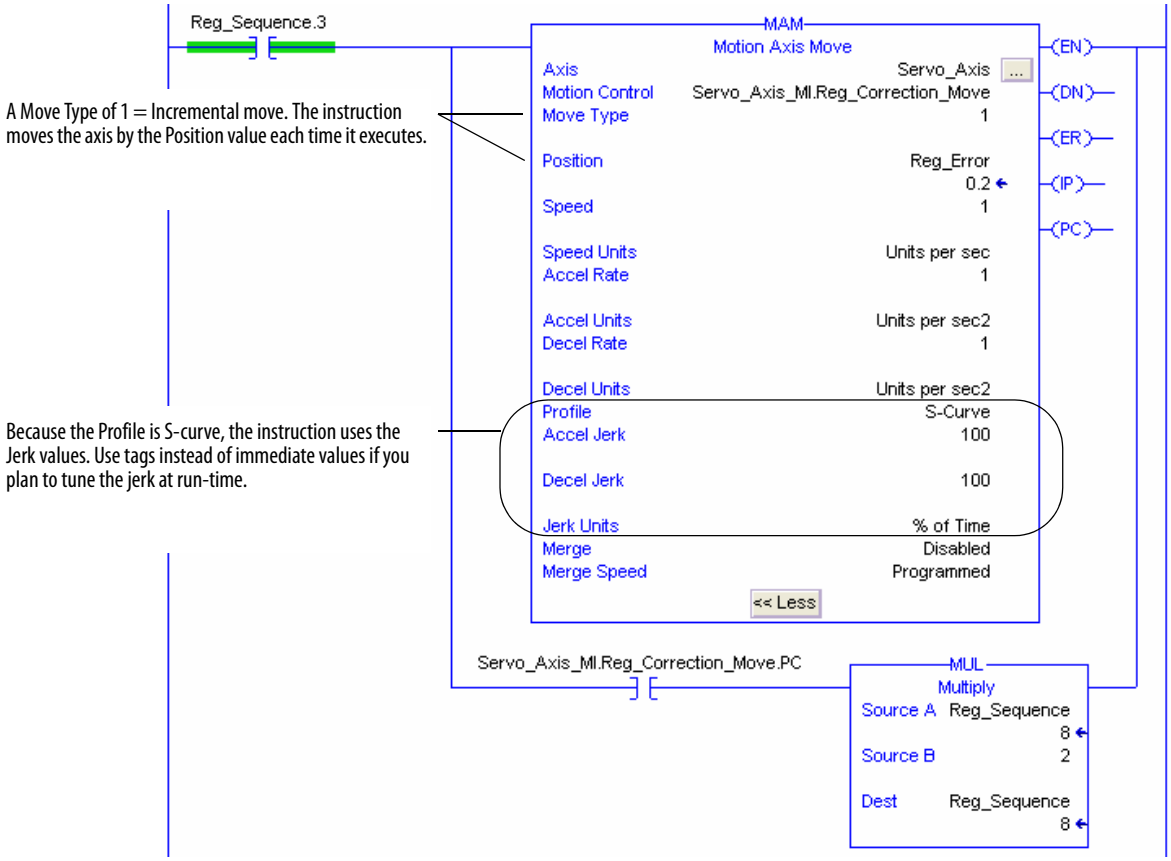
MAM Example

Move—Relay Ladder

This example uses the bit pattern of *Reg_Sequence* to step through the logic.

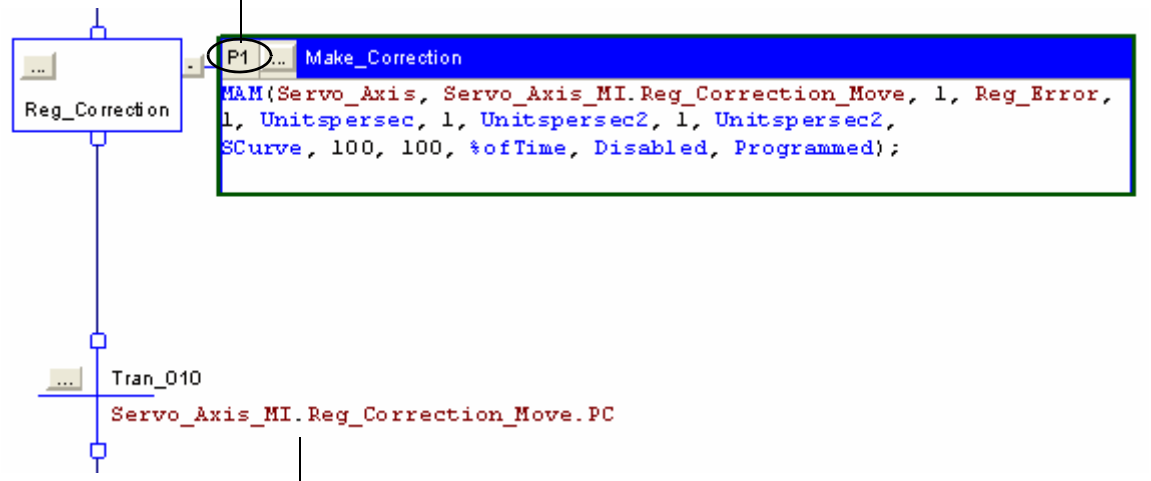
When *Reg_Sequence.3* turns on

1. Move *Servo_Axis* the distance of *Reg_Error*.
2. When the move is complete, multiply *Reg_Sequence* by 2. This turns off bit 3 of *Reg_Sequence* and turns on bit 4.



Move—Structured Text

When the SFC starts the *Reg_Correction* step, move *Servo_Axis* the distance of *Reg_Error*. The P1 qualifier limits this to the first scan of the step.



The PC bit for the move turns on when the move is complete. The SFC leaves the step when the PC bit turns on.

Motion Axis Gear (MAG)

The MAG instruction enables electronic gearing between two axes at a specified ratio. Electronic gearing allows any physical axis to be synchronized to the actual or command position of another physical axis at a precise ratio.

ATTENTION: The maximum velocity, acceleration, or deceleration limits established during axis configuration do not apply to electronic gearing.

Operands

The MAG instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

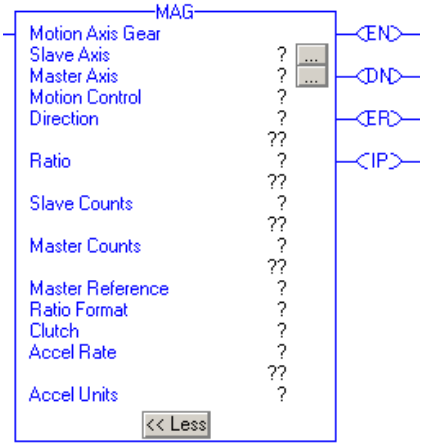


Table 45 - MAG Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Slave axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Master axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_CONSUMED AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	The axis that the slave axis follows.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Table 45 - MAG Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Direction	DINT	Immediate Tag	The relative direction that the Slave axis tracks the Master Axis. Select one of following: 0 = slave axis moves in the same direction as the master axis When Same is selected or entered as the Direction, the slave axis moves in its positive direction at the specified gear ratio when the master axis moves in its positive direction and vice-versa. 1 = slave axis moves in the opposite direction of its current direction When Opposite is selected or entered as the Direction, the slave axis moves in its negative direction at the specified gear ratio when the master axis moves in its positive direction and vice-versa. 2 = slave axis reverses from current or previous When Unchanged is selected or entered as the Direction, the gear ratio can be changed while preserving the current gearing direction (same or opposite). This is useful when the current direction is not known or not important. 3 = slave axis to continue its current or previous direction When Reverse is selected or entered as the Direction, the current direction of the electronic gearing is changed from same to opposite or from opposite to same. This is very useful for winding applications where the gear ratio must be reversed at each end of the wind.
Ratio	REAL	Immediate Tag	Signed Real value establishing the gear ratio in Slave User Units per Master User Unit.
Slave counts	DINT or SINT	Immediate Tag	Integer value representing slave counts used in specifying a Fractional gear ratio.
Master counts	DINT or SINT	Immediate Tag	Integer value representing master counts used in specifying a Fractional gear ratio.
Master reference	DINT	Immediate	Sets the master position reference to either Command position or Actual position. 0 = Actual – slave axis motion is generated from the current position of the master axis as measured by its encoder or other feedback device. 1 = Command – slave axis motion is generated from the desired or commanded position of the master axis.
Ratio format	DINT	Immediate	The desired ratio specification format. 0 = real gear ratio 1 = integer fraction of slave encoder counts to master encoder counts
Clutch	DINT	Immediate	When Clutch is enabled, motion control ramps the slave axis up to gearing speed at the instruction's defined Acceleration value. If not enabled, the Slave axis immediately locks onto the Master axis. If the Master Axis is currently moving this condition results in an abrupt uncontrolled acceleration event of the Slave Axis which can cause the axis to fault. 0 = disabled 1 = enabled
Accel rate	DINT	Immediate Tag	Acceleration rate of the Slave Axis in % or Acceleration Units. It is applied when the Clutch feature is enabled.
Accel units	DINT	Immediate	The units used to display the Acceleration value. 0 = units per sec^2 1 = % of maximum acceleration

Structured Text

MAG(SlaveAxis,MasterAxis, MotionControl,Direction, Ratio,SlaveCounts, MasterCounts,MasterReference,RatioFormat,Clutch,AccelRate, AccelUnits);

The operands are the same as those for the relay ladder MAG instruction.

See Table 45, MAG Relay Ladder Operand Descriptions on page 123.

For the operands that require you to select from available options, enter your selection as described in Table 46.

Table 46 - MAG Structured Text Operand Descriptions

This Operand	Has These Options That You	
	Enter as Text	Or Enter as a Number
MasterReference	actual command	0 1
RatioFormat	real fraction_slave_master_counts	0 1
Clutch	disabled enabled	0 1
AccelUnits	unitspersec^2 %ofmaximum	0 1

MAG MOTION_INSTRUCTION Structure

Table 47 - MAG Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis gear has been successfully initiated.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared if either superseded by another Motion Gear Axes command, or terminated by a stop command, merge, shutdown, or servo fault.

Description

The Motion Axis Gear (MAG) instruction provides electronic gearing between any two axes in a specified direction and at a specified ratio. Electronic gearing provides a direct edge-to-edge lock between the two axes—no maximum velocity, acceleration, or deceleration limits are used. The speed, acceleration, and deceleration of the slave axis is completely determined by the motion of the master axis and the specified gear ratio.

When called, the specified Slave Axis is geared to the Master Axis at the specified Ratio (for example, 1.345) or Slave Counts to Master Counts (for example, 1:3). The MAG instruction supports specification of the gear ratio in one of two different formats, Real or Fractional, as determined by the Ratio Format input selection. The direction of Slave Axis motion relative to the Master Axis is defined by a very flexible Direction input parameter. The gearing direction can be explicitly set as the Same or Opposite or set relative to the current gearing direction as Reverse or Unchanged.

The value for Ratio is sign sensitive. The Master Reference selection allows gearing input to be derived from either the Actual or Command position of the Master Axis. When the instruction's Clutch capability is activated the gearing instruction commands the slave axis to accelerate or decelerate at a controlled rate before Locking on to the master axis by using the instructions Acceleration value much like the clutch of a car.

Select or enter the desired Master Axis, Slave Axis, and Direction and enter a value or tag variable for the desired ratio. If an axis is dimmed (gray) or not shown in the Slave Axis pop-up menu, the physical axis is not defined for Servo operation.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for servo operation. Use the Tag Editor to create and configure a new axis.

Electronic gearing remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic gearing motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

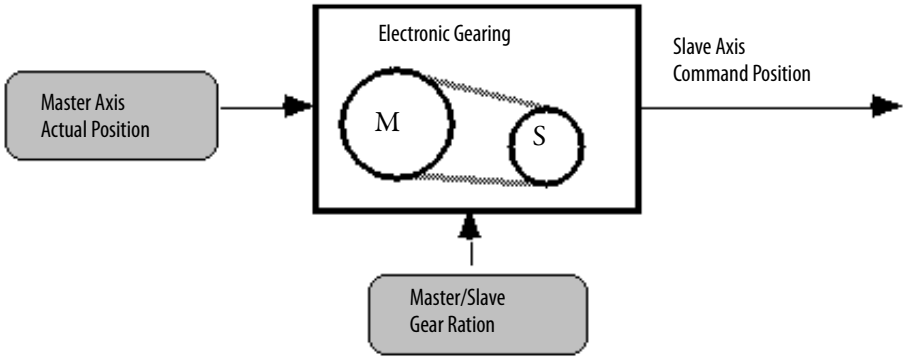
For more information, see Structured Text Programming on page 359.

MAG Programming Guidelines

Follow these are guidelines when programming an MAG instruction.

Slaving to the Actual Position

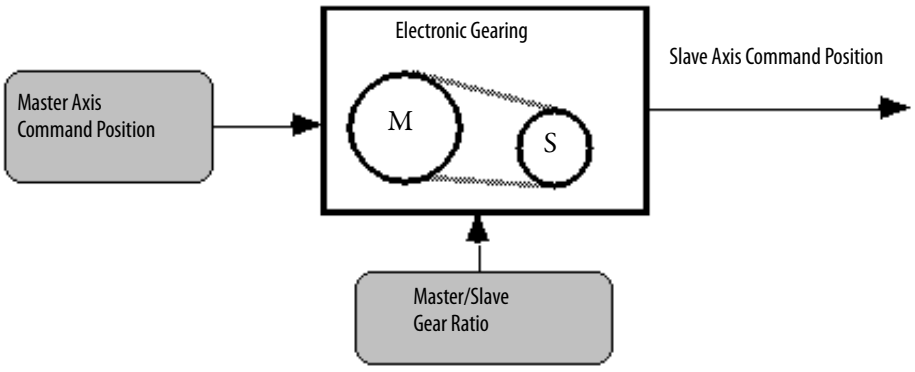
When Actual Position is entered or selected as the Master Reference source, the slave axis motion is generated from the actual position of the master axis as shown in the following graphic.



Actual position is the current position of a physical axis as measured by the axis encoder. This is the only valid selection when the master axis' Axis Type is configured as Feedback Only.

Slaving to the Command Position

When Command Position is entered or selected as the Master Reference source, the slave axis motion is generated from the command position of the master axis as shown in the following graphic.



Command position (valid only when the master axis' Axis Type is configured as Servo) is the current desired or commanded position for the master axis.

Because the command position does not incorporate any associated following error, external position disturbances, or quantization noise, it is a more accurate and stable reference for gearing. When gearing to the command position of the master, the master axis must be commanded to move to cause any motion on the slave axis.

Specifying a Gear Ratio

There are two types of gear ratios, real number and fraction:

- Real Number Gear Ratios

  When Ratio Format is selected or entered as Real, the gear ratio is specified as a real number or tag variable with a value between 0.00001 and 9.99999 (inclusive) representing the desired ratio of slave axis position units to master axis position units. A gear ratio expressed this way is easy to interpret because it's defined in the axes' configured position units.

- Fraction Gear Ratios

  When Ratio Format is selected or entered as Fraction, the gear ratio is specified as a pair of integer numbers or tag variables representing the ratio between the number of slave axis feedback counts and the number of master axis feedback counts. See The Tag variable Builder earlier in this manual for information on tag variables.

  IMPORTANT The Conversion Constant entered as part of the axis configuration procedure is not used when the Ratio Format for the MAG instruction is specified as a Fraction.

  If your gear ratio cannot be exactly expressed as a real number with a maximum of five digits to the right of the decimal point, use Fraction as the Ratio Format.

  Specifying the gear ratio as a fraction allows the direct implementation of irrational gear ratios (such as 1/3) with no accumulated positioning errors or round off. Because the master and slave count values do not use the axis conversion constants and because they are integers, the actual gear ratio relationship between the slave and master axes exactly match the specified ratio.

  For example, the irrational gear ratio of 1/3 can be equivalently specified as 1 slave count to 3 master counts, 10 slave counts to 30 master counts, 3 slave counts to 9 master counts.

Clutch Function

By using the clutch feature, you avoid the uncontrolled acceleration or deceleration that results when electronic gearing is enabled while the master axis is moving. The clutch feature can also be used to merge gear ratio changes on-the-fly, even changes in direction.

The motion controller automatically ramps the slave axis to the speed implied by the master axis at the new ratio and/or direction. The operation of the clutch ramp generator has no affect on jog or move processes that might be in progress on the slave axis.

Changing Master Axis

The master axis for electronic gearing can be changed at any time, even while gearing is currently enabled. However, because its possible to have electronic gearing enabled on more than one axis at a time, if a Servo master axis and slave axis are reversed, the axes become cross-coupled and unexpected motion can result.

For example, if you are gearing Axis 0 to Axis 1 (defined as a Servo axis) and then want to change to gearing Axis 1 to Axis 0, you must first disable gearing (StopType=Gear) on Axis 0. This is because specifying Axis 1 as the slave axis with Axis 0 as the master axis does not automatically disable Axis 0 from being a slave axis with Axis 1 as the master axis.

See Motion Axis Stop (MAS) on page 82 for more information.

Moving While Gearing

An incremental MAM instruction can be used for the slave axis (or master axis if the Axis Type is configured as Servo) while the electronic gearing is enabled. This is particularly useful to accomplish phase advance/retard control. The incremental move distance can be used to eliminate any phase error between the master and the slave, or to create an exact non-zero phase relationship. Incremental MAM instruction can also be used in conjunction with electronic gearing to compensate for material slip.

Normally a gear ratio of 1 is used with phase adjustment. A 1:1 ratio ensures that the computed phase error does not change before performing the move to correct it. Electronic gearing is not normally used with absolute moves, because the ultimate endpoint is not predictable.

Arithmetic Status Flags

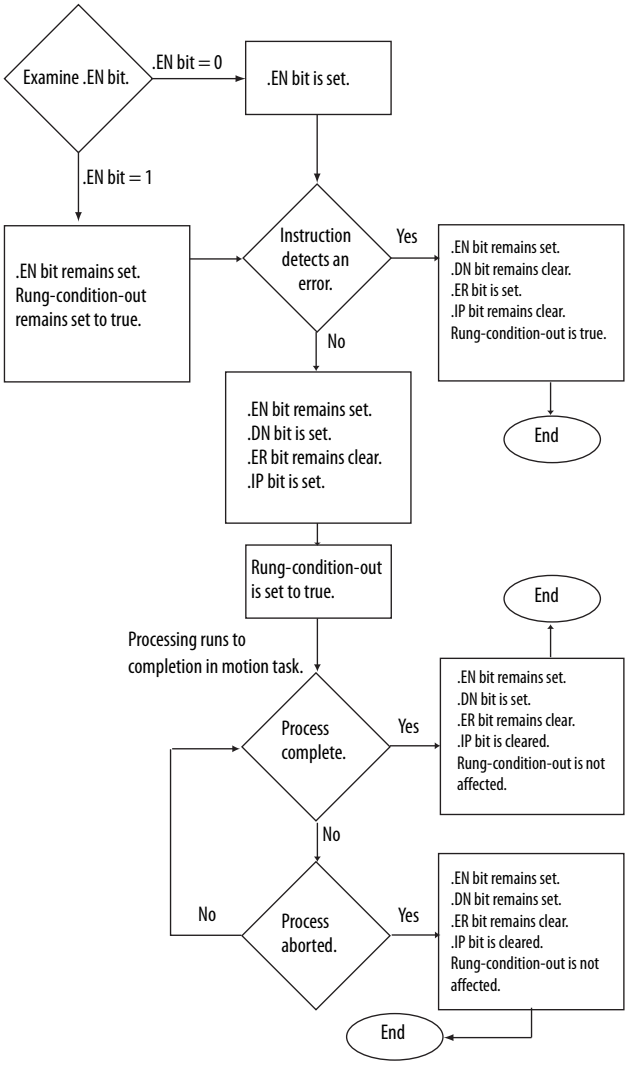
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAG Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, .ER, and .IP bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, and .IP bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MAG Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAG Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions.

Use Extended Error Codes (EXERR) for more information about an error.

Table 48 - MAG Extended Error Codes Description

If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
11	Varies	Slave or master axis is not configured.	Configure the axis.	
			1	Slave
			2	Master
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed
			4	Ratio
54	-1	The coordinate system has a Maximum Deceleration of 0.	Click the Properties for the coordinate system axis and set a Maximum Deceleration.	
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	1. Open the Properties for the axis. 2. Use the EXERR value to see which axis has the Maximum Deceleration of 0. 3. The axis that you are jogging has a deceleration rate of 0. 4. Click the ellipsis button next to the offending axis to access the Axis Properties screen. 5. Click the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Click the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.	

MAG Changes to Status Bits

If the Clutch check box is NOT checked, execution of the MAG instruction simply sets the Gear Status bit to True.

Bit Name	State	Meaning
GearingStatus	TRUE	Axis is Gearing

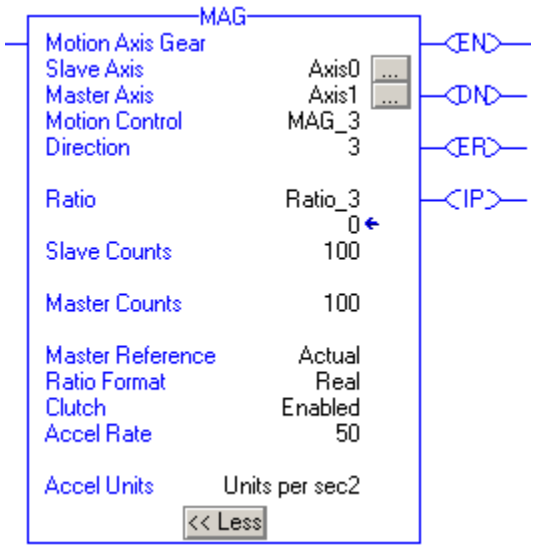
If the Clutch check box is checked, execution of the MAG instruction sets the Gearing Lock Status bit to True when the clutching process completes.

Bit Name	State	Meaning
Gearing Lock Status	TRUE	Axis has finished Clutch and locked in.
GearingStatus	TRUE	Axis is Gearing

MAG Example

When the input conditions are true, the controller provides electronic gearing between *axis2* and *axis1*.

Relay Ladder



Structured Text

MAG(Axis0,Axis1,MAG_3,3,Ratio_3,0,100,100,Actual,Real, Enabled,50,Unitspersec2);

Motion Change Dynamics (MCD)

Use the Motion Change Dynamics (MCD) instruction to selectively change the speed, acceleration rate, or deceleration rate of a move profile or a jog profile in process.

IMPORTANT	If you put a value of 0 for the Speed, any motion commands in process will terminate and will not function as a feedhold, limiting the current axis to zero speed.

Operands

The MCD instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

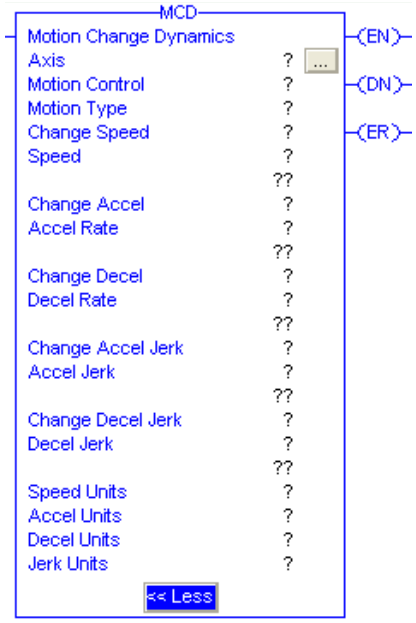


Table 49 - MCD Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Motion type	DINT	Immediate	Motion profile (jog or move) to change. 0 = jog 1 = move

Table 49 - MCD Relay Ladder Descriptions (Continued)

Operand	Type	Format	Description
Change speed	DINT	Immediate	Set to enable a change of speed. 0 = no 1 = yes
Speed	REAL	Immediate Tag	The new Speed to move the axis in % or Speed Units.
Change Accel	DINT	Immediate	Set to enable an acceleration change. 0 = no 1 = yes
Accel rate	REAL	Immediate Tag	The acceleration rate of the axis in % or Acceleration units.
Change Decel	DINT	Immediate	Set to enable a deceleration change. 0 = no 1 = yes
Decel rate	REAL	Immediate Tag	The deceleration rate of the axis in % or Deceleration units. The axis could overshoot its target position if you reduce the deceleration while a move is in process.
Change Accel Jerk	SINT, INT, or DINT	Immediate	0 = No 1 = Yes
Accel Jerk	SINT, INT, DINT, or REAL	Immediate Tag	You **must** always enter a value for the Accel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. • Accel Jerk is the acceleration jerk rate for the axis. Use this value to get started. Accel Jerk = 100 (% of Time) Jerk Units = 2
Change Decel Jerk	DINT	Immediate	0 = No 1 = Yes
Decel Jerk	SINT, INT, DINT, or REAL	Immediate Tag	You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. • Decel Jerk is the deceleration jerk rate for the coordinate system. Use these values to get started. Decel Jerk = 100 (% of Time) Jerk Units = 2
SpeedUnits	DINT	Immediate	Units used to display the Speed value. 0 = units per sec 1 = % of maximum speed 2 = Units per Master Unit

Table 49 - MCD Relay Ladder Descriptions (Continued)

Operand	Type	Format	Description
Accel units	DINT	Immediate	Units used to display the Acceleration value. 0 = units per sec^2 1 = % of maximum acceleration 2 = Units per MasterUnits^2
Decel units	DINT	Immediate	Units used to display the Deceleration value. 0 = units per sec^2 1 = % of maximum deceleration 2 = Units per MasterUnits^2
Jerk Units	DINT	Immediate	0 = Units per sec^3 1 = % of Maximum 2 = % of Time 3 = Percent of time 4 = Units per Master Units^3 5 = Percent of Time-Master Driven

Structured Text

MCD(Axis,MotionControl,MotionType,ChangeSpeed, Speed,ChangeAccel,AccelRate,ChangeAccelJerk,AccelJerk,ChangeDecelJerk, ChangeDecel,DecelRate,SpeedUnits,AccelUnits, DecelUnits,JerkUnits);

The operands are the same as those for the relay ladder MCD instruction.

See Table 49, MCD Relay Ladder Descriptions on page 133.

For the operands that require you to select from available options, enter your selection as described in Table 50.

Table 50 - MCD Structure Text Operand Descriptions

This Operand	Has These Options That You	
	Enter as Text	Or Enter as a Number
Axis	No enumeration	Tag
MotionControl	No enumeration	Tag
MotionType	jog move	0 1
ChangeSpeed	no yes	0 1
Speed	No enumeration	Immediate, Tag
ChangeAccel	no yes	0 1
AccelRate	No enumeration	Immediate, Tag
ChangeDecel	no yes	0 1
Decel Rate	No enumeration	Immediate, Tag
ChangeAccelJerk	No enumeration	0 = No 1 = Yes

Table 50 - MCD Structure Text Operand Descriptions (Continued)

This Operand	Has These Options That You	
	Enter as Text	**Or Enter as a Number**
AccelJerk	No enumeration	Immediate, Tag You **must** always enter a value for the Accel operand. This instruction only uses the value if the Profile is configured as S-curve. Use this value to get started. Accel Jerk = 100
ChangeDecelJerk	No enumeration	0 = No 1 = Yes
DecelJerk	No enumeration	Immediate, Tag You **must** always enter a value Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Use this value to get started. Decel Jerk = 100
SpeedUnits	unitspersec %ofmaximum	0 1
AccelUnits	unitspersec^2 %ofmaximum	0 1
DecelUnits	unitspersec^2 %ofmaximum	0 1
JerkUnits	Unitspersec^3 %ofmaximum %oftime	0 1 2

MCD MOTION_INSTRUCTION Structure

Table 51 - MCD Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) BIt 29	It is set when axis change dynamics has been successfully initiated. The instruction execution completes in a single scan, and the DN bit is set immediately.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MCD instruction changes the speed of trapezoidal profile moves on-the-fly and the speed, acceleration, and deceleration of trapezoidal profile jogs on-the-fly. Choose the desired physical axis and type of motion and enter values or tag variables for the Speed, Accel, and Decel. Speed, acceleration, and deceleration values can be entered as percentages of the current maximum configured value or directly in the configured speed or acceleration units of the axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for servo operation. Use the Tag Editor to create and configure a new axis.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if one of these occurs:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

For more information, see Analyzing Axis Motion on page 331.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  For more information, see Structured Text Programming on page 359.

Programming Guidelines

These are programming guidelines for the MCD instruction.

Changing Move Dynamics

When a Motion type of Move is entered or chosen, the speed, acceleration, and/or deceleration of a Move in progress can be changed to the specified value. The speed change occurs at the specified acceleration rate if the new speed is higher than the current speed or at the specified deceleration rate if the new speed is lower than the current speed.

Pausing Moves

The MCD instruction can be used to temporarily pause a move in progress by changing its speed to zero. Use another MCD instruction with a non-zero speed value to complete the move as originally specified.

Changing Jog Dynamics

When a Motion type of Jog is entered or chosen, the speed, acceleration, and/or deceleration of a Jog in progress can be changed to the specified value. The speed change occurs at the specified acceleration rate if the new speed is higher than the current speed or at the specified deceleration rate if the new speed is lower than the current speed.

Changing Axis Direction

You can execute an MCD instruction during another instruction's execution to change an axis' speed and the direction in which that axis moves. There are two MCD instruction operands that affect the speed and directional movement of an axis.

- Change Speed - Must be set to 1, so that MCD instruction changes the speed of the axis. If this value is 0, no change of speed occurs.
- Speed - The new Speed to move the axis in % or Speed Units.
  - A positive value in the Speed operand causes a positive directional move on the axis.
  - A negative value in the Speed operand causes a negative directional move on the axis.

EXAMPLE If your program is executing a Motion Axis Jog (MAJ) instruction, with its Direction operand set to 1, the axis moves in reverse. If you were then required to reduce that jogged move's speed, you would execute an MCD instruction while the MAJ instruction was in process. The MCD instruction's Change Speed operand must be 1. Then, depending on the MCD instruction's Speed operand value, the effect on the axis' directional movement varies.

Setting the Speed operand to a new positive value, for example, 10 % of max, would result in the axis changing directions, because the Speed operand is positive, dynamically over-riding the MAJ instruction's reverse movement. To maintain the same direction in this example, that is, reverse, the MCD instruction's Speed operand value needs to be a negative value, for example, -10 % of max.

Supporting the Motion Drive Start (MDS) Instruction

The MCD instruction supports the Motion Drive Start (MDS) instruction. However, the MCD instruction has no affect on the DirectVelocityControlStatus Command feature because the Motion Planner takes only the value from the Direct Command Velocity Attribute and sums it into the axis output before sending the total command to the drive. After all acceleration and deceleration have been planned, the MCD instruction has no affect on the feature.

Arithmetic Status Flags

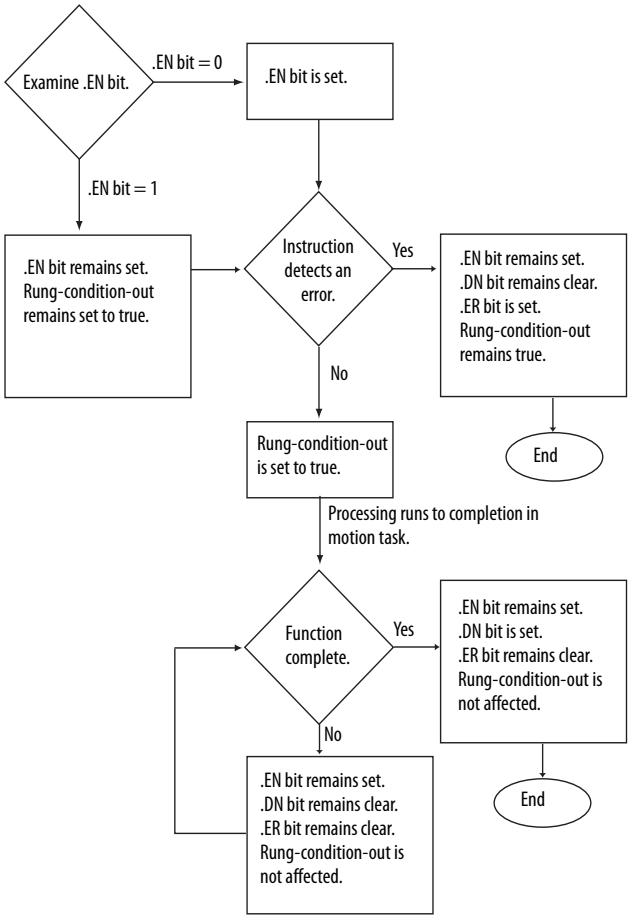
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MCD Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MCD Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MCD Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions.

Use Extended Error Codes (EXERR) for more information about an error.

Table 52 - MCD Extended Error Codes Description

If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed
			4	Ratio
54	-1	The coordinate system has a Maximum Deceleration of 0.	Click the Properties for the coordinate system axis and set a Maximum Deceleration.	
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	1. Open the Properties for the axis. 2. Use the EXERR value to see which axis has the Maximum Deceleration of 0. 3. The axis that you are jogging has a deceleration rate of 0. 4. Click the ellipsis button next to the offending axis to access the Axis Properties screen. 5. Click the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Click the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.	

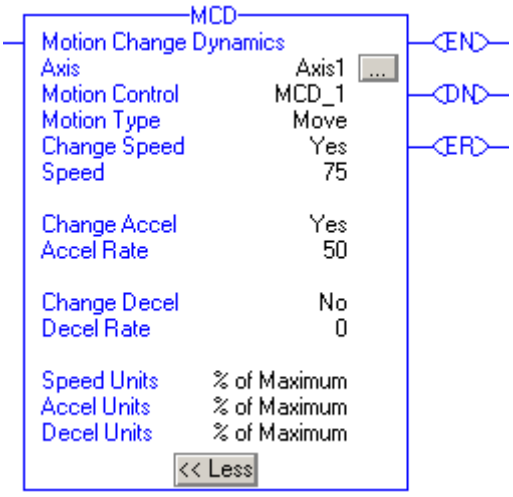
MCD Changes to Status Bits

The MCD instruction makes no changes to status bits.

MCD Example

When the input conditions are true, the controller changes the speed, acceleration, or deceleration rate of a move profile or jog profile in progress for *axis1*.

Relay Ladder



Structured Text

MCD(Axis1,MCD_1,Move,Yes,75,Yes,50,No,0,%ofmaximum, %ofmaximum,%ofmaximum);

Motion Redefine Position (MRP)

Use the Motion Redefine Position (MRP) instruction to change the command or actual position of an axis.The value specified by Position is used to update the Actual or Command position of Axis. The position redefinition can be calculated on an Absolute or Relative basis. If Absolute is selected the Position value is assigned to the current Actual or Command position. If Relative is selected the Position value is added as a displacement to the current Actual or Command position.

The process of redefining the current axis position has no affect on motion in progress as the instruction preserves the current servo following error during the redefinition process. As a result, axis position can be redefined on-the-fly without disturbing axis motion.

Operands

The MRP instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

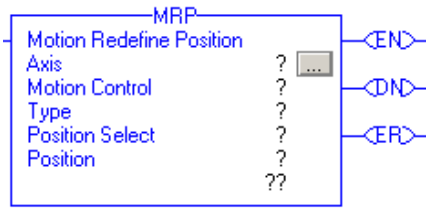


Table 53 - MRP Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction parameters.
Type	BOOLEAN	Immediate	The way you want the redefinition operation to work. 0 = absolute 1 = relative
Position select	BOOLEAN	Immediate	Choose what position to perform the redefinition operation on. 0 = actual position 1 = command position
Position	REAL	Immediate Tag	The value to use to change the axis position to or offset to current position.

Structured Text

MRP(Axis,MotionControl,Type,PositionSelect,Position);

The operands are the same as those for the relay ladder MRP instruction. Enter your selection for the operands that require you to select from available options.

Table 54 - MRP Structured Test Operand Descriptions

This Operand	Has These Options That You	
	Enter as Text	Or Enter as a Number
Type	absolute relative	0 1
PositionSelect	actual command	0 1

MRP MOTION_INSTRUCTION Structure

Table 55 - MRP Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' position action been successfully redefined.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MRP instruction directly sets the actual or command position of the specified axis to the specified absolute or relative position. No motion is caused by this instruction—the current axis position is simply redefined. Select or enter the desired Axis, Type, Position Selection, and enter a value or tag variable for the desired New Position.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRP instruction can be used while the axis is moving as well as when it is at rest. MRP is used to redefine position 'on-the-fly' for certain registration, slip compensation, and recalibration applications.

Executing the Instruction

To successfully execute a MRP instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errs.

You will get error 85, Homing not allowed when a redefine position is in process, if any of the motion planner instructions are executed while MRP is in progress. The motion instructions included are: MAM, MAJ, MCLM, MCCM, MATC, MAPC, MDAC, and MDCC

IMPORTANT The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  For more information, see Structured Text Programming on page 359.

Programming Guidelines

These are programming guidelines for the MRP instruction.

Absolute Mode

When Absolute is selected or entered as the MRP Type, the New Position specifies the new absolute position of the axis. No motion occurs—the current axis position (actual or command) is simply redefined to be the specified new position.

If software overtravel limits are used (refer to Motion Axis Object specification for more information on software overtravel configuration), the new position must be between the Max Positive and Max Negative Travel configuration values. Otherwise a software overtravel fault is generated when the instruction is executed.

ATTENTION: If software overtravel limit checking is in effect, execution of an MRP in Absolute mode can invalidate the current Max Positive and Max Negative Travel limits in the absolute sense. Exercise caution when redefining the absolute position of an axis that has travel limits.

Absolute and relative mode MRP instructions have the same effect when the axis is not moving. When the axis is moving, however, absolute mode introduces a position error equal to the motion of the axis during the time it takes to execute the MRP instruction and assign the new position. Relative mode does not introduce this error and guarantees an exact correction independent of axis speed or position.

Relative Mode

When Relative is selected or entered as the MRP Type, the New Position value is used to offset the current position of the axis. No motion occurs—the current axis position (actual or command) is simply redefined to be the current position plus the specified new position.

In relative mode, axis position is redefined in such a way that no position errors are introduced if the axis is moving. It is particularly useful for unwinding axis position under program control rather than using the built-in rotary axis feature.

Absolute and relative mode MRP instructions have the same effect when the axis is not moving. When the axis is moving, however, absolute mode introduces a position error equal to the motion of the axis during the time it takes to execute the MRP instruction and assign the new position. Relative mode does not introduce this error and guarantees an exact correction independent of axis speed or position.

Actual Position

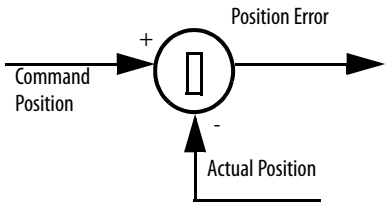
When Actual is selected or entered as the MRP Position Selection, the New Position is directly applied to the actual position of the physical axis. The command position of the axis is also adjusted along with the new actual position to preserve any position error which exists. This ensures that there is no unexpected motion of the axis when the positions are redefined.

See the Integrated Motion on the Ethernet/IP Network Reference Manual, publication MOTION-RM003, for more discussion of command position, actual position, and position error.

Command Position

When Command is selected or entered as the MRP Position Selection, the New Position is directly applied to the command position of the servo or imaginary axis. Because Feedback Only axes do not have a command position, always choose Actual from the Position menu for Master Only axes. The actual position of servo axes is also adjusted along with the new command position to preserve any position error which exists. This ensures that there is no unexpected motion of the axis when the positions are redefined.

Command position is the desired or commanded position of a servo as generated by any previous motion instructions. Actual position is the current position of a physical or virtual axis as measured by the encoder or other feedback device. Position error is the difference between these two and is used to drive the motor to make the actual position equal to the command position. This figure shows the relationship of these three positions.



Arithmetic Status Flags

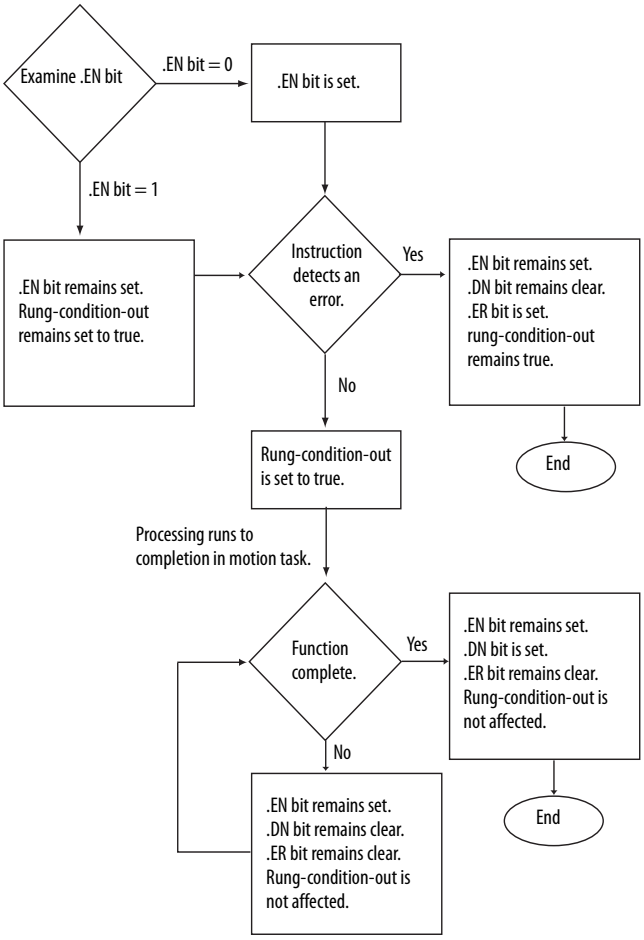
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MRP Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, and .ER, bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MRP Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MRP Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MRP instruction receives a Servo Message Failure (12) error message.

Table 56 - MRP Extended Error Code Description

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive. (sercos).

Extended Error Codes for the Parameter Out of Range (13) error code work a little differently. Rather than having a standard enumeration, the number that appears for the Extended Error code refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MRP instruction, an extended error code of 4 would refer to the Position operand's value. You would then have to check your value with the accepted range of values for the instruction.

MRP Changes to Status Bits

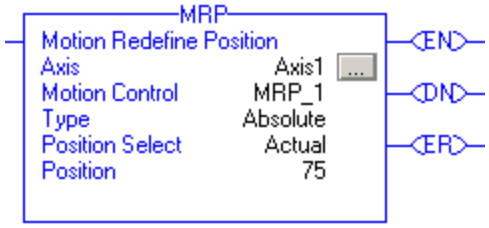
If the axis has been homed prior to executing the MRP instruction, the HomedStatus bit is set. The HomedStatus bit is cleared when the MRP instruction is executed. This indicates the axis position is no longer referenced to the home position.

If the axis has been homed by using the absolute home procedure, the AbsoluteReferenceStatus is set. HomedStatus can also be set is the axis has not been subject to a power cycle. AbsoluteReferenceStatus should be cleared when the MRP is executed. This indicates the axis position is no longer referenced to the absolute home position.

MRP Example

When the input conditions are true, the controller changes the position of *axis1*.

Relay Ladder



Structured Text

```
MRP(Axis1,MRP_1,Absolute,Actual,75);
```

Motion Calculate Cam Profile (MCCP)

The Motion Calculate Cam Profile (MCCP) instruction calculates a cam profile based on an array of cam points. An array of cam points can be established programmatically or by use of the Logix Designer Cam Profile Editor. Each cam point in the cam array consists of a slave position value, a master position(position cam) or time(time cam) value, and an interpolation type (linear or cubic). The resulting cam profile can be used by a Motion Axis Position Cam (MAPC) instruction or Motion Axis Time Cam (MATC) instruction to govern the motion of a slave axis according to master position or time.

Operands

The MCCP instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

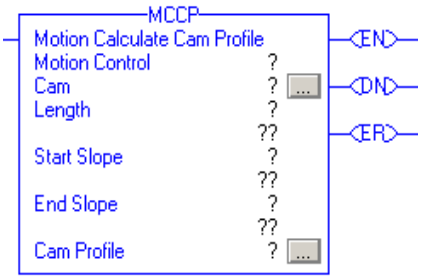


Table 57 - MCCP Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.
Cam	CAM	Array	Tag name of the cam array used to compute the cam profile. The numerical array index indicates the starting cam element in the array used in the cam profile calculation. Ellipsis launches Cam Profile Editor.
Length	SINT, DINT	Immediate Tag	Determines the number of cam elements in the array used in the cam profile calculation.
Start Slope	REAL	Immediate Tag	This is the boundary condition for the initial slope of the profile. It is valid only for a cubic first segment and is used to specify a slope through the first point.
End Slope	REAL	Immediate Tag	This is the boundary condition for the ending slope of the profile. It is valid only for a cubic last segment and is used to specify a slope through the last point.
Cam Profile	CAM_PROFILE	Array	Tag name of the calculated cam profile array used as input to MAPC and MATC instructions. Only the zero array element ([0]) is allowed for the Cam Profile array. Ellipsis launches Cam Profile Editor.

Structured Text

MCCP(MotionControl,Cam, Length,StartSlope,EndSlope,CamProfile);

The operands are the same as those for the relay ladder MCCP instruction. For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]).

MCCP MOTION_INSTRUCTION Structure

Table 58 - MCCP Motion_Instruction Structure Description

Enumerations	Description
.EN (Enable) Bit 31	The enable bit is set when the rung transitions from false-to-true and stays set until the done bit is set and the rung goes false.
.DN (Done) Bit 29	The done bit is set when the calculate cam instruction has been successfully executed and the Cam Profile array calculated.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if the cam array is of an illegal length.

Description

The MCCP instruction computes a cam profile based on a given set of points in a specified cam array. The Cam array elements consist of slave (yp) and master (xp) point pairs as well as an interpolation type. Because there is no association with a specific axis position or time, the x and y point values are unitless. The interpolation type can be specified for each point as either 'linear' or 'cubic'.

The resultant cam profiles generated by this instruction can be used by subsequent Motion Axis Position Cam (MAPC) on page 156 or Motion Axis Time Cam (MATC) on page 178 camming instructions to provide complex motion of a slave axis with respect to either a master axis position or with respect to time.

Because cam profiles can be directly calculated by the Logix Designer Cam Profile Editor, the main purpose of the MCCP instruction is to provide a method for calculating cam profiles in real-time based on programmatic changes to the corresponding cam arrays.

Executing the Instruction

IMPORTANT The MCCP instruction execution completes in a single scan. This instructions should therefore be placed in a separate task to avoid impacting user program scan time.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  For more information, see Structured Text Programming on page 359.

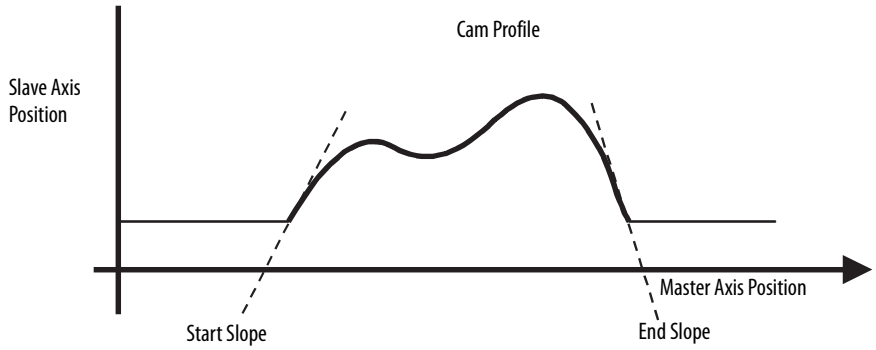
Linear and Cubic Interpolation

The resultant calculated cam profiles are fully interpolated. This means that if the current master position or time does not correspond exactly with a point in the cam array used to generate the cam profile, the slave axis position is determined by linear or cubic interpolation between adjacent points. In this way, the smoothest possible slave motion is provided. The MCCP instruction accomplishes this by calculating coefficients to a polynomial equation that determines slave position as a function of master position or time.

Start Slope and End Slope

To facilitate a smooth entry into and exit from a cubic cam profile, slope control is provided. The Start Slope and End Slope parameters determine the initial rate of change of the slave relative to the master. These values are used in the cubic spline calculations performed on the cam array. Figure 5 shows the master slave slope relationship.

Figure 5 - Start and End Slope



The default values for Start Slope and End Slope are 0 to facilitate a smooth start and end to the cam profile from rest. However, if the axis is already camming, an appropriate non-zero Start Slope can be specified to match the End Slope of the currently executing cam, to seamlessly blend the two cam profiles together.

The Start Slope and End Slope values are not applicable when starting or ending the cam profile with linear interpolation.

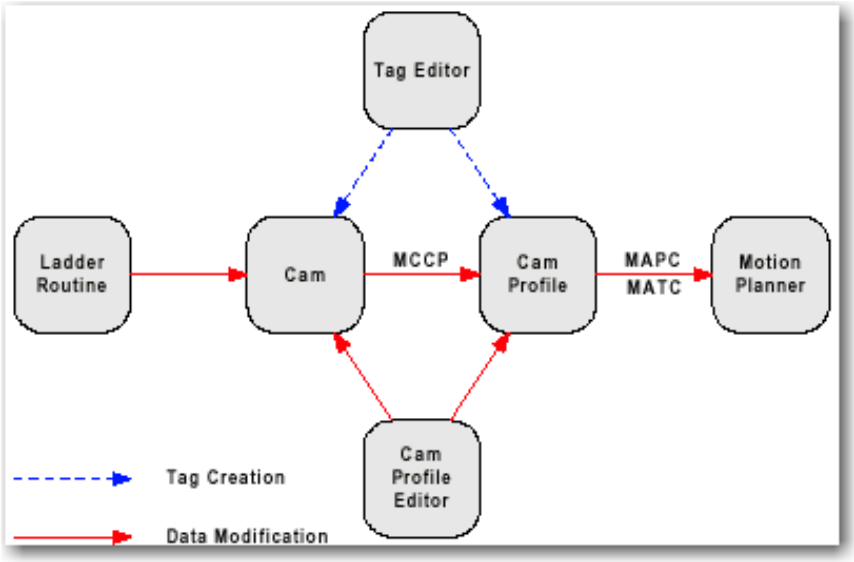
Specifying and Calculating the Cam Profile

To execute an MCCP instruction, a Cam Profile array tag must also be created. Cam Profile array tags can be created by the Logix Designer tag editor or the MAPC/MATC instructions by using the built-in Cam Profile Editor.

The data within the Cam Profile array can be modified at compile time by using the Cam Profile Editor, or at run-time with the MCCP instruction. In the case of run-time changes, a Cam array must be created in order to use the MCCP instruction.

The status parameter is used to indicate that the Cam Profile array element has been calculated. If execution of a camming instruction is attempted by using any uncalculated elements in a cam profile, the MAPC or MATC instructions error. The type parameter determines the type of interpolation applied between this cam array element and the next cam element.

Figure 6 - Cam Operation Diagram



The Status member of the first element in the cam profile array is special and used for data integrity checks. For this reason, the MCCP must always specify the cam profile with the starting index set to 0. This first cam profile element Status member can have the following values.

Table 59 - MCCP Status Variable Descriptions

Status Variables	Description
0	Cam profile element has not been calculated
1	Cam profile element is being calculated
2	Cam profile element has been calculated
n	Cam profile element has been calculated and is currently being used by (*n*-2) MAPC or MATC instructions

Before calculating a cam profile on a specified axis, the MCCP instructions first checks if the cam profile array has been calculated by checking the value of the first cam profile element's Status member. If the Status value is either 0 or 2, the MCCP proceeds with the calculation of the cam profile.

When the cam profile array has been completely calculated, the MCCP instruction sets the first cam profile element's Status value to 'being calculated', or 1, and then sets the Status value of all other cam profile elements to 'being calculated'. As the calculation proceeds, individual cam profile members' Status values are set to 'calculated', or 2. When all elements in the cam profile array have been calculated, the first cam profile element's Status value is also set to 'calculated'.

However, if an MCCP instruction is executed with an initial cam profile Status value of 1, then the cam profile is currently being calculated by another MCCP instruction, and the MCCP instruction errors. If the Status value is >2, then the cam profile is being actively used by an MAPC or MATC instruction process, and the MCCP instruction errs.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MCCP Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MCCP Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are not specific enough to help pinpoint the problem. When the MCCP instruction receives an Illegal Cam Length (26) error message to let it know that the length input parameter does not correspond to what the instruction expects, the corresponding Extended Error code provides the number of cams in the Cam Tag provided to the instruction. When the MCCP instruction receives an Illegal Cam Profile Length (27) error message to let it know that the length input parameter does not correspond to what the instruction expects, the corresponding Extended Error code provides the number of cam points the instruction is attempting to generate.

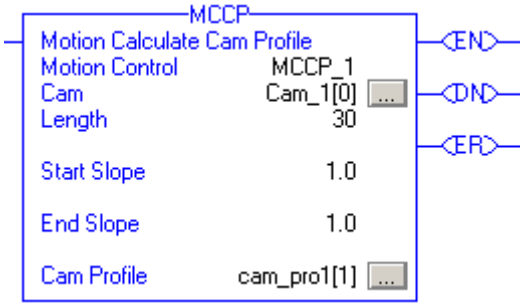
MCCP Changes to Status Bits

The MCCP instruction makes no changes to the status bits.

MCCP Example

Relay Ladder

When the input instructions are true, the controller changes the speed, acceleration, or deceleration rate of a move profile or jog profile in progress for axis_1.



Structured Text

MCCP(MCCP_1,Cam_1[0],30,1.0,1.0,cam_pro1[1]);

Motion Axis Position Cam (MAPC)

The Motion Axis Position Cam (MAPC) instruction provides electronic camming between any two axes according to the specified cam profile. When an MAPC instruction is executed, the specified Slave Axis is synchronized to the designated Master Axis by using a position cam profile established by the Logix Designer Cam Profile Editor, or by a previously executed Motion Calculate Cam Profile (MCCP) instruction.

The MAPC instruction executes a position cam profile set up by a previous Motion Calculate Cam Profile (MCCP) instruction or, alternatively, by the Logix Designer Cam Profile Editor. Position cams, in effect, provide the capability of implementing non-linear "electronic gearing" relationships between two axes. No maximum velocity, acceleration, or deceleration limits are used. The speed, acceleration, and deceleration of the slave axis are completely determined by the motion of the master axis and the designated cam profile derived from the associated cam table.

ATTENTION: The maximum velocity, acceleration, or deceleration limits established during axis configuration do not apply to electronic camming.

Operands

The MAPC instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

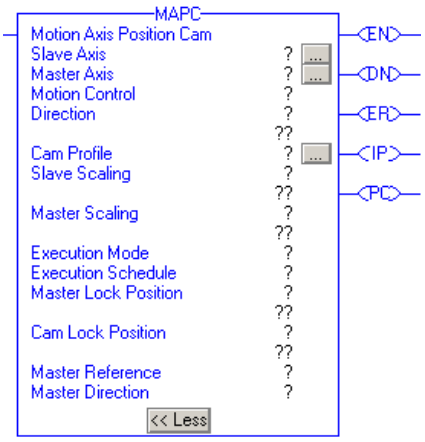


Table 60 - MAPC Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Slave Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	The name of the axis that the cam profile is applied to. Ellipsis launches Axis Properties dialog box.
Master Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_CONSUMED AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	The axis that the slave axis follows according to the cam profile. Ellipsis launches Axis Properties dialog box. If Pending is selected as the Execution Schedule, then Master Axis is ignored.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.
Direction	DINT, SINT	Immediate Tag	Cams can be configured to add or subtract their incremental contribution to the slave axis command position. Control over this behavior is via the Direction parameter and is relative to the direction of the slave axis to the master axis. 0 = Same The slave axis position values are in the same sense as the master's. When Same is selected or entered as the Direction for the MAPC instruction, the slave axis position values computed from the cam profile are added to the command position of the slave axis. This is the most common operation, as the profile position values are used just as entered in the original cam table. That is, consecutive increasing profile values result in axis motion in the positive direction and vice-versa. 1 = Opposite The slave axis position values are in the opposite sense of the master's. When Opposite is selected or entered as the Direction, the slave axis position values computed from the cam profile are subtracted from the command position of the slave axis. Thus, axis motion is in the opposite direction from that implied by the original cam table. That is, consecutive increasing profile values result in axis motion in the negative direction and vice-versa. Or relative to the current or previous camming direction. 2 = Reverse The current or previous direction of the position cam is reversed on execution. When executed for the first time with Reverse selected, the control defaults the direction to Opposite. 3 = Unchanged This allows other cam parameters to be changed without altering the current or previous camming direction (same or opposite). This is useful when the current direction is not known or is not important. When executed for the first time with Unchanged selected, the control defaults the direction to Same.

Table 60 - MAPC Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Cam Profile	CAM_PROFILE	array	Tag name of the calculated cam profile array used to establish the master/slave position relationship. Only the zero array element ([0]) is allowed for the Cam Profile array. Ellipsis launches Cam Profile Editor.
Slave Scaling	REAL	Immediate Tag	Scales the total distance covered by the slave axis through the cam profile.
Master Scaling	REAL	Immediate Tag	Scales the total distance covered by the master axis through the cam profile.
Execution Mode	DINT	Immediate	Execution modes of Once or Continuous can be selected to determine how the cam motion behaves when the master position moves beyond the start and end points of the profile defined by the original cam table. When Continuous mode is selected, the specified cam profile, once started, is executed indefinitely. With continuous operation, the profile's master and slave positions are `unwound' when the position of the master axis moves outside the profile range, causing the cam profile to repeat. To generate smooth continuous motion by using this technique, however, care must be taken in designing the cam points of the cam table to ensure that there are no position, velocity, or acceleration discontinuities between the start and end points of the calculated cam profile. 0 = Once Cam motion of slave axis starts only when the master axis moves into the range defined by the start and end points of the cam profile. When the master axis moves beyond the defined range of the profile cam motion on the slave axis stops and the Process Complete bit is set. Slave motion does not resume if the master axis moves back into the cam profile range. When the master axis moves outside the range of the profile, cam motion on the slave axis stops and the Process Complete bit of the MAPC instruction is set. Note that, contrary to the current S Class practice, slave motion does not resume when and if the master moves back into the profile range specified by the start and end points. 1 = Continuous Once started the cam profile is executed indefinitely. This feature is useful in rotary applications where it is necessary that the cam position run continuously in a rotary or reciprocating fashion. 2 = Persistent When the Master Axis moves beyond the defined range, cam motion on the Slave Axis stops and the PositionCamLockStatus bit is cleared. Slave motion resumes in the opposite direction when the Master Axis reverses and moves back into the cam profile range, at which time the PositionCamLockStatus bit is set.
Execution Schedule	DINT	Immediate	Selects the method used to execute the cam profile. Control over the MAPC instruction's execution is via the Execution Schedule parameter. Options are: 0 = Immediate – The slave axis is immediately locked to the master axis and the position camming process begins. The Master Lock Position parameter is irrelevant. 1 = Pending – lets you blend a new position cam execution after an in process position cam is finished. When Pending is selected the following parameters are ignored: Master Axis, Master Lock Position, and Master Reference. 2 = Forward only – the cam profile starts when the master position crosses the Master Lock Position in the forward direction. 3 = Reverse only – the cam profile starts when the master position crosses the Master Lock Position in the reverse direction. 4 = Bidirectional – the cam profile starts when the master position crosses the Master Lock Position in either direction.
Master Lock Position	REAL	Immediate Tag	When the Master Offset = 0.0, the Master Lock Position is the Master axis absolute position where the slave axis locks to the master axis. If the Master Offset is X, then the Slave axis locks to the Master axis at the absolute master position value of Master Lock Position –X. For example, assume a Master Lock Position =50 and a Master Offset Move =10. Assume the Master axis move (MAM) and Master offset move (MOM) start at the same time, then the Slave locks to the Master at an absolute Master axis position of 40. This, in effect shifts the Cam profile of 10 units to the left. If Pending is selected as the Execution Schedule value, then Master Lock Position is ignored.

Table 60 - MAPC Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Cam Lock Position	REAL	Immediate Tag	This determines the starting location in the cam profile.
Master Reference	DINT	Immediate	Sets the master position reference to either Command position or Actual position. If Pending is selected for the Execution Schedule value, then Master Reference is ignored. 0 = Actual – slave axis motion is generated from the current position of the master axis as measured by its encoder or other feedback device. 1 = Command – slave axis motion is generated from the desired or commanded position of the master axis.
Master Direction	DINT	Immediate	This determines the direction of the master axis that generates slave motion according to the cam profile. Options are: 0 = Bidirectional – default, slave axis can track the master axis in either direction. 1 = Forward only – slave axis tracks the master axis in the forward direction of the master axis. 2 = Reverse only – slave axis tracks the master axis in the opposite direction of the master axis.

Structured Text

MAPC(SlaveAxis,MasterAxis,MotionControl,Direction,CamProfile,SlaveScaling,MasterScaling,ExecutionMode,ExecutionSchedule,MasterLockPosition,CamLockPosition,MasterReference,MasterDirection);

The operands are the same as those for the relay ladder MAPC instruction. For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]).

Enter your selection for the operands that require you to select from available options.

Table 61 - MAPC Structured Text Operand Descriptions

This Operand	Has These Options That You	
	Enter as Text	Or Enter as a Number
ExecutionMode	once continuous persistent	0 1 2
ExecutionSchedule	immediate pending forwardonly reverseonly bidirectional	0 1 2 3 4
MasterReference	actual command	0 1
MasterDirection	bidirectional forwardonly reverseonly	0 1 2

MAPC MOTION_INSTRUCTION Structure

Table 62 - MAPC Motion_Instruction Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN (Done) Bit 29	The done bit is set when the axis position cam instruction is successfully initiated.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if the axis is not configured.
.IP (In Process) Bit 26	The In Process bit is set on positive rung transition and cleared if either superseded by another Motion Axis Position Cam command, or terminated by a stop command, merge, shutdown, or servo fault.
.PC (Process Complete) Bit 27	It is cleared on positive rung transition and set, in `once' Execution mode, when the position of the master axis leaves the master position range defined by the currently active cam profile.

Description

The Motion Axis Position Cam (MAPC) instruction provides electronic camming between any two axes according to the specified cam profile. The direction of Slave Axis motion relative to the Master Axis is defined by a flexible Direction input parameter. To accurately synchronize the slave axis position to master axis position, an Execution Schedule setting and an associated Master Lock Position can be specified for the master axis. When the master axis travels past the Master Lock Position in the direction specified by the Execution Schedule parameter, the slave axis is locked to the master axis position according to the specified Cam Profile beginning at the Cam Lock Position.

The cam profile can also be configured to execute Immediately or Pending completion of a currently executing position cam profile via the Execution Schedule parameter. The cam profile can be executed once, continuous, or in a persistent mode, by specifying the desired Execution mode.

The Master Reference selection allows camming input from the master to be derived from either the Actual or Command position of the Master Axis. To support applications which require unidirectional motion, a “slip clutch” feature is available which prevents the slave from “backing-up” when the master axis reverses direction. This feature is controlled by the Master Direction parameter.

To support applications which require unidirectional motion, a ‘slip clutch’ feature is available, which prevents the slave from ‘backing-up’ when the master axis reverses direction. Master and Slave Scaling functionality can be used to scale slave motion based on a standard cam profile without having to create a new cam table and calculate a new cam profile.

Linear and Cubic Interpolation

Position cams are fully interpolated. This means that if the current Master Axis position does not correspond exactly with a point in the cam table associated with the cam profile, the slave axis position is determined by linear or cubic interpolation between the adjacent points. In this way, the smoothest possible slave motion is provided.

Each point in the Cam array that was used to generate the Cam Profile can be configured for linear or cubic interpolation.

Electronic camming remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic camming motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

Executing the Instruction

IMPORTANT	The MAPC instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process (.IP) bit are set immediately. The In Process (.IP) bit remains set until the initiated PCAM process completes, is superseded by another MAPC instruction, terminated by a Motion Axis Stop command, Merge operation, or Servo Fault Action. The Process Complete bit is cleared immediately when the MAPC executes and sets when the cam process completes when configured for `Once' Execution mode.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  For more information, see Structured Text Programming on page 359.

Specifying the Cam Profile

To execute a MAPC instruction, a calculated Cam Profile data array tag must be specified. Cam Profile array tags can be created by the Logix Designer tag editor or the MAPC instruction by using the built-in Cam Profile Editor, or by executing an Motion Calculate Cam Profile (MCCP) instruction on an existing Cam array.

The data within the Cam Profile array can be modified at compile time by using the Cam Profile Editor, or at run-time with the Motion Calculate Cam Profile (MCCP) instruction. In the case of run-time changes, a Cam array must be created in order to use the MCCP instruction.

Refer to the Motion Calculate Cam Profile (MCCP) instruction specification for more detail on converting Cam arrays.

The Status member of this Cam Profile is used is used to indicate that the corresponding Cam Profile array element has been calculated. If execution of a camming instruction is attempted with any uncalculated elements in a cam profile, the instruction errors. The type parameter determines the type of interpolation applied between this cam array element and the next cam element, (for example, linear or cubic).

The Status member of the first element in the cam profile array is special and used for data integrity checks. For this reason, the MAPC must always specify the cam profile with the starting index set to 0.

This first cam profile element Status member can have the following values.

Table 63 - MAPC Status Member Value Descriptions

Status Value	Description
0	Cam profile element has not been calculated
1	Cam profile element is being calculated
2	Cam profile element has been calculated
n	Cam profile element has been calculated and is currently being used by (*n*-2) MAPC or MATC instructions

Before starting a cam on a specified axis, the MAPC instructions checks if the cam profile array has been calculated by checking the value of the first cam profile element's Status member. If Status is 0 or 1 then the cam profile has not been calculated yet and the MAPC instruction errors. If the cam profile array has been completely calculated (Status > 1), the instruction then increments the Status member indicating that it is in use by this axis.

When the cam completes, or terminates, the Status member of the first cam profile array element is decremented to maintain track of the number of cams actively by using the associated cam profile.

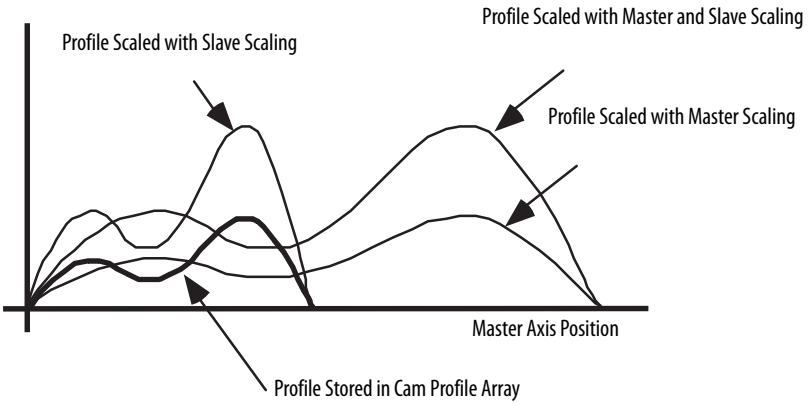
Scaling Position Cams

A position cam profile can be scaled in both the master dimension and slave dimension when it is executed. This scaling feature is useful to allow the stored cam profile to be used to determine the general **form** of the motion profile. The scaling parameters are then used to define the total master or slave travel over which the profile is executed, as shown in Figure 7. In this way, one standard cam profile can be used to generate a whole family of specific cam profiles.

When a cam profile array is specified by an MAPC instruction, the master and slave values defined by the cam profile array take on the position units of the master and slave axes respectively. By contrast, the Master and Slave Scaling

parameters are unitless values that are simply used as multipliers to the cam profile.

Figure 7 - Cam Profile Array


Profile Scaled with Master and Slave Scaling
Profile Scaled with Slave Scaling
Profile Scaled with Master Scaling
Master Axis Position
Profile Stored in Cam Profile Array

By default, both the Master Scaling and Slave Scaling parameters are set to 1. To scale a position cam profile, enter a Master Scaling or Slave Scaling value other than 1.

Note that increasing the master scaling value of a cam profile **decreases** the velocities and accelerations of the profile, while increasing the slave scaling value **increases** the velocities and accelerations of the profile. To maintain the velocities and accelerations of the scaled profile approximately equal to those of the unscaled profile, the master scaling and slave scaling values should be equal. For example, if the slave scaling value of a profile is 2, the master scaling value should also be 2 to maintain approximately equal velocities and accelerations during execution of the scaled position cam.

ATTENTION: Decreasing the Master Scaling value or increasing the Slave Scaling value of a position cam increases the required velocities and accelerations of the profile. This can cause a motion fault if the capabilities of the drive system are exceeded.

MAPC Execution Schedule

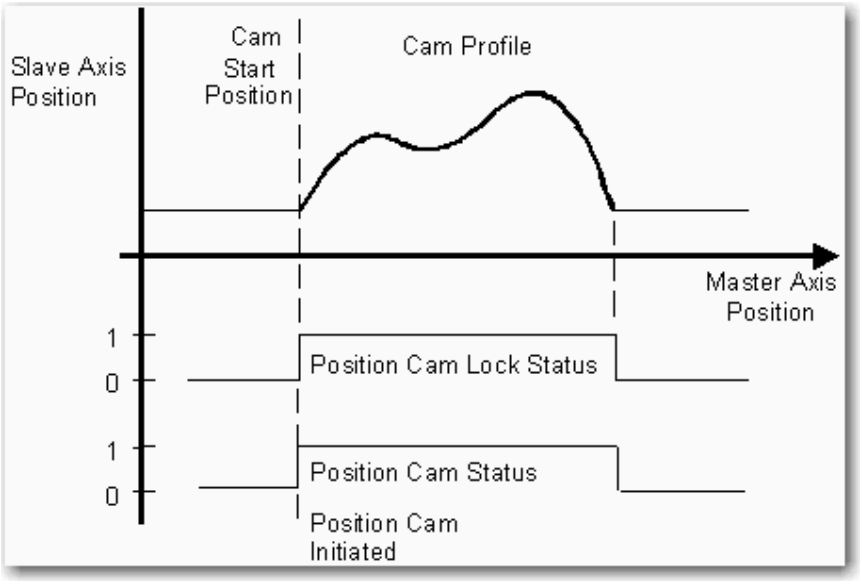
Control over the MAPC instruction's execution is via the Execution Schedule parameter.

Immediate Execution

By default, the MAPC instruction is scheduled to execute Immediately. In this case, there is no delay to the enabling of the position camming process and the Master Lock Position parameter is irrelevant. The slave axis is immediately locked to the master axis beginning at the Cam Lock Position of the specific cam profile.

As illustrated in Figure 8, when the MAPC instruction is executed, the camming process is initiated on the specified slave axis and the Position Cam Status bit in the slave axis' Motion Status word is set. If the Execution Schedule parameter is set to Immediate, the slave axis is immediately locked to the master according to the specified Cam Profile. This is indicated by the fact that the Position Cam Lock Status bit for the specified slave axis is also set.

Figure 8 - Immediate Execution



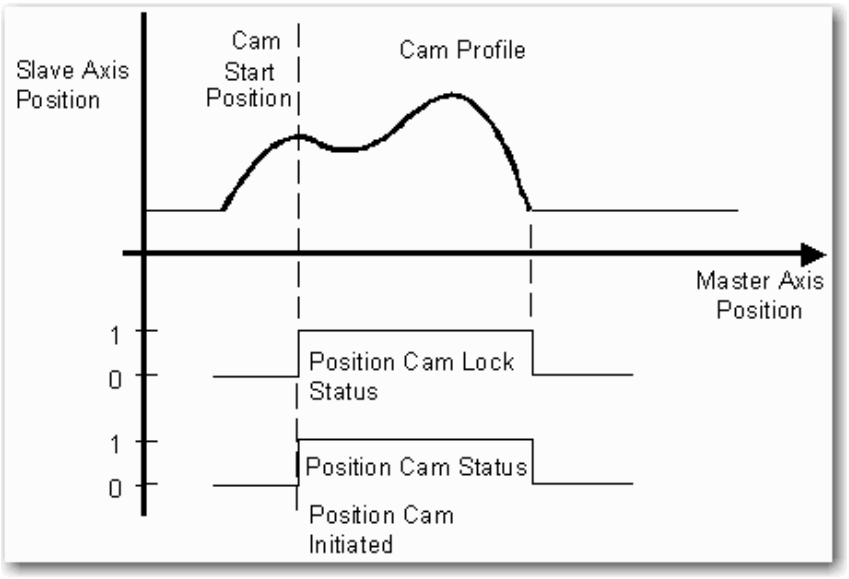
If the Execution Schedule of an MAPC instruction is set to Immediate and a position cam profile is currently in process, the MAPC instruction errs. This is true even when the axis is waiting to lock onto the master axis.

Changing the Cam Lock Position with Immediate Execution Schedule

The Cam Lock Position parameter of the MAPC instruction determines the starting location within the cam profile when the slave locks to the master. Typically, the Cam Lock Position is set to the beginning of the cam profile as shown in Figure 8. Because the starting point of most cam tables is 0, the Cam Lock Position is typically set to 0. Alternatively, the Cam Lock Position can be set to any position within the master range of the cam profile. If a Cam Lock Position is specified that is out of this range, the MAPC instruction errors.

Figure 9 shows the effect of specifying a Cam Lock Position value other than the starting point of the cam table, in this case, a position within the cam profile itself. Care must be taken not to define a Cam Start Point that results in a velocity or acceleration discontinuity to the slave axis if the master axis is currently moving.

Figure 9 - Changing the Cam Lock Position

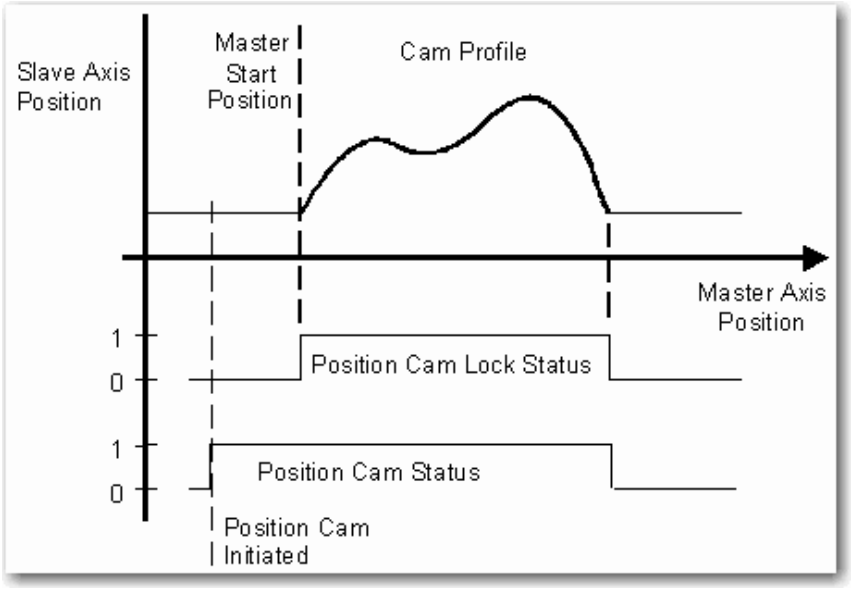


Forward Only, Reverse Only, or Bidirectional Execution

In the case where the Execution Schedule parameter of the instruction is set to Forward Only, Reverse Only, or Bidirectional, the slave axis is not locked to the master until the master axis satisfies the specified condition. In this case, the master axis is monitored by the camming process to determine when the master axis passes the specified Master Lock Position in the specified direction. In a rotary axis configuration, this lock criterion is still valid, independent of the turns count.

IMPORTANT If the position reference of the master axis is redefined (for example an MRP instruction) after the MAPC instruction executes but before the lock condition is satisfied, the cam profile generator monitors the master axis based on the absolute position reference system in effect prior to the redefine position operation.

Figure 10 - Forward Only, Reverse Only, or Bidirectional Execution



When the absolute position of the master axis passes the specified Master Lock Position in the specified direction, the Position Cam Status bit of the Motion Status word for specified slave axis is set. Slave axis motion is then initiated according to the specified cam profile starting at the specified Cam Lock Position of the cam profile. From this point on, only the **incremental change** in the master axis position is used to determine the corresponding slave axis position from the defined cam profile. This is important for applications where the master axis is a rotary axis because the position cam is then unaffected by the position unwind process.

When the master axis moves out of the range defined by the cam profile (assuming Execution mode configured for Once), both the Position Cam Lock Status and the Position Cam Status bits of the Motion Status word are cleared. This Motion Status bit condition indicates that the cam process has completed.

This fact is also reflected in the bit leg behavior of the associated MAPC instruction, PC bit set and IP bit clear.

After position cam motion is started when the master axis passes the specified Master Lock Position in either the Forward Only or Reverse Only direction, the master axis can change direction and the slave axis reverses accordingly.

If an MAPC instruction is executed on a slave axis that is already actively position camming, an Illegal Dynamic Change error is generated (error code 23). The only exception for this is if the Execution Schedule is specified as 'pending'.

Pending Cam Execution

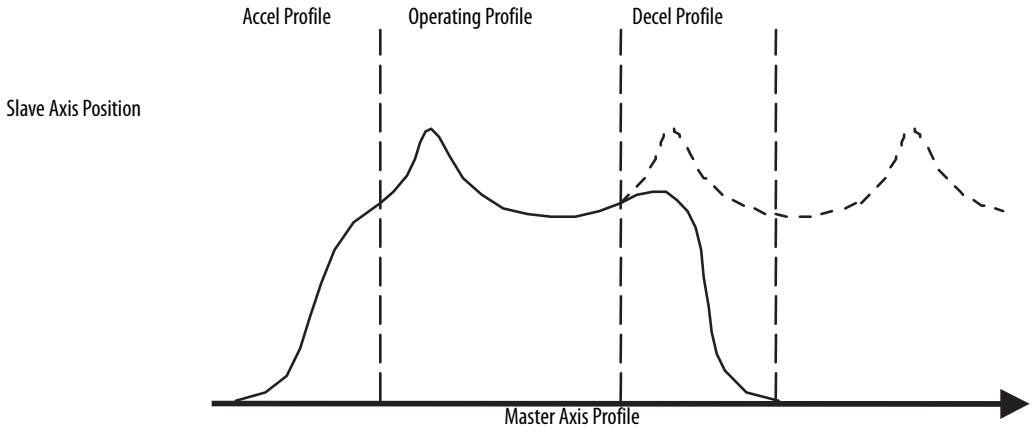
The transition from one cam profile to another occurs when the operational cam profile reaches the last point in its table. At this point, the pending cam profile table goes into operation.

Alternatively, the MAPC instruction's execution can be deferred pending completion of a currently executing position cam. An Execution Schedule selection of Pending can thus be used to seamlessly blend two position cam profiles together without stopping motion.

As with the continuous mode of operation, the blending points should not have large value differences. Large differences can result in dramatic acceleration and deceleration changes. Blending points are the start and end points for the continuous cam profile. In other words, the points are the last point of the operating cam profile and first point of pending cam profile.

The Pending execution feature is particularly useful in applications like high-speed packaging when a slave axis must be locked onto a moving master axis and accelerate by using a specific profile to the proper speed. When this acceleration profile is done, it must be smoothly blended into the operating profile, which is typically executed continuously. To stop the slave axis, the operating profile is smoothly blended into a deceleration profile such that the axis stops at a known location as shown in graphic Pending Cam Execution.

Figure 11 - Pending Cam Execution



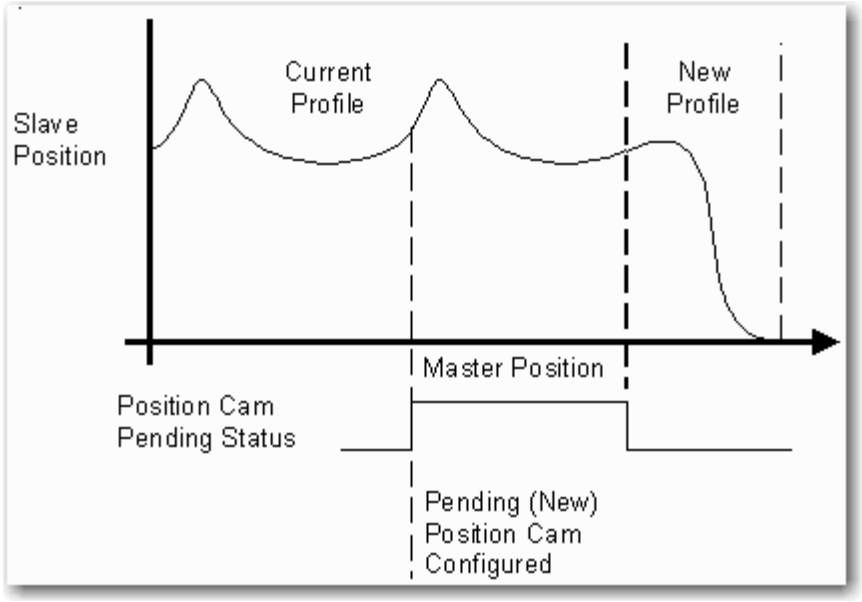
By executing the position cam profile as a Pending cam profile while the current profile is still executing, the appropriate cam profile parameters are set up ahead of time. This makes the transition from the current profile to the pending profile seamless; synchronization between the master and slave axes is maintained. To ensure smooth motion across the transition, however, the profiles must be designed such that no position, velocity, or acceleration discontinuities exist between the end of the current profile and the start of the new one. This is done by using the Logix Designer Cam Profile Editor.

Once a pending position cam instruction has been executed, the new cam profile takes effect automatically (and becomes the current profile) when the master axis passes through either the start or end point of the current profile. If the current cam is configured to execute once, the new profile is initiated at the completion of the pass through the current cam profile and the PC bit of the currently active MAPC instruction is set. If the current cam is configured to execute continuously, the new profile is initiated at the completion of the current pass through the current cam profile and the IP bit of the currently active MAPC instruction is cleared. The motion controller keeps track of the master axis and slave axis positions relative to the first profile at the time of the change and uses this information to maintain synchronization between the profiles.

If an Execution Schedule of Pending is selected without a corresponding position cam profile in progress, the MAPC instruction executes but no camming motion occurs until another MAPC instruction with a non-pending Execution Schedule is initiated. This allows pending cam profiles to be preloaded prior to executing the initial cam. This method addresses cases where immediate cams would finish before the pending cam could be reliably loaded.

After a Pending position cam has been configured, the Position Cam Pending Status bit of the Motion Status word for the specified slave axis is set to 1 (true). When the pending (new) profile is initiated and becomes the current profile, Position Cam Pending Status bit is immediately cleared as shown in Figure 12.

Figure 12 - Pending Position Cam



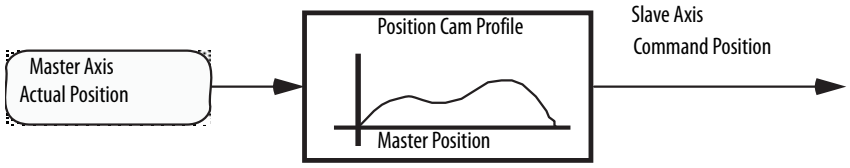
Master Position Reference

The Master Reference parameter determines the master position source to link to the cam generator. This source can be actual position or command position of the master axis. Smoother motion is derived from command position but in some cases, for example when a physical axis is not controlled by a ControlLogix motion module, actual position is the only practical option.

Slaving to the Actual Position

When Actual Position is entered or selected as the Master Reference source, the slave axis motion is generated from the actual position of the master axis as shown in Figure 13.

Figure 13 - Slaving to the Actual Position

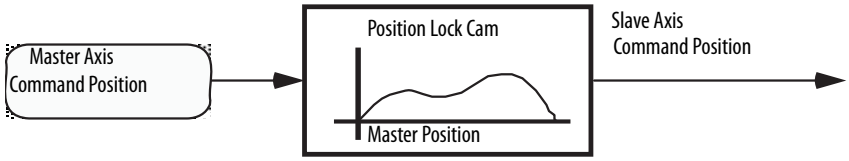


Actual position is the current position of the master axis as measured by its encoder or other feedback device. This is the default selection and the only selection when the master Axis Type is configured as Feedback Only because it is often necessary to synchronize the actual positions of two axes.

Slaving to the Command Position

When Command Position is entered or selected as the Master Reference source, the slave axis motion is generated from the command position of the master axis as shown in Figure 14.

Figure 14 - Slaving to the Command Position



Command position (available only when the master axis' Axis Type is a Servo or Virtual axis) is the desired or commanded position of the master axis.

Because the command position does not incorporate any associated following error or external position disturbances, it is a more accurate and stable reference for camming. When camming to the commanded position of the master, the master axis could be **commanded** or moved manually while in disabled state to cause corresponding motion on the slave axis.

Master Direction

Normally, the Master Direction parameter is set to Bidirectional (default). However, when Forward Only is selected for Master Direction, the slave axis tracks the master axis in the forward direction of the master axis. When Reverse Only is selected, the slave axis tracks the master axis in the reverse direction of the master axis. If the master axis changes direction, the slave axis does not reverse direction, but stays where it was when the master reversed. This Unidirectional feature of position cams is used to provide an electronic slip clutch, which prevents the cam motion generator from moving backward through the cam profile if the master reverses direction.

When the master axis again reverses, resuming motion in the desired direction, the slave axis 'picks up' again when the master reaches the position where it initially reversed. In this way, the slave axis maintains synchronization with the master while motion in the wrong direction is inhibited. This is especially useful where motion in a certain direction can cause physical damage to the machine or to the product.

Moving While Camming

Motion Axis Moves can be performed while camming to provide sophisticated phase and offset control while the slave axis is running.

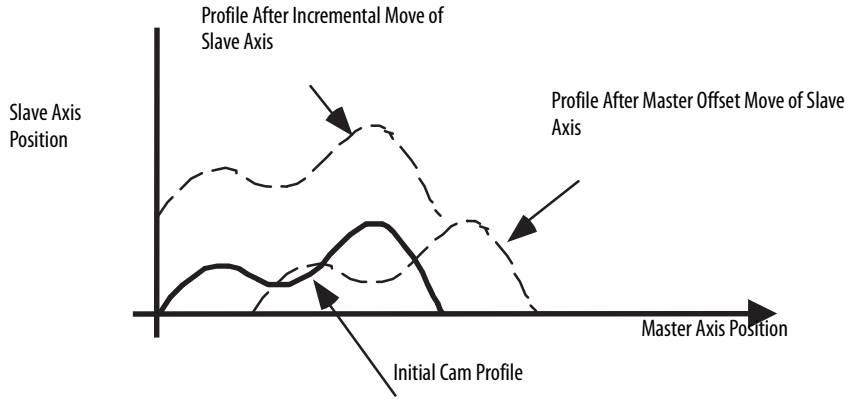
Incremental Moves

An Incremental Motion Axis Move (MAM) on page 111 can be used on the slave axis (or master axis if configured for Servo operation) while the position cam is operating. This is particularly useful to accomplish phase advance/retard control. The incremental move distance can be used to eliminate any phase error between the master and the slave, or to create an exact phase relationship.

Master Offset Moves

A MAM instruction can also be used while the position cam is operating to shift the master reference position of the cam on the fly. Unlike an incremental move on the slave axis, a master offset move on the slave axis shifts the cam profile relative to the master axis, as shown in Figure 15.

Figure 15 - Master Offset Move


Profile After Incremental Move of Slave Axis
Slave Axis Position
Profile After Master Offset Move of Slave Axis
Master Axis Position
Initial Cam Profile

When the MAPC instruction (except pending) is initiated, the corresponding active Master Offset Move is disabled and the corresponding Master Offset, Strobe Offset, and Start Master Offset are reset to zero. In order to achieve the master reference position shift, the MAM instruction must be initiated after the MAPC is initiated.

See the Motion Axis Move (MAM) on page 111 instruction for more information on Master Offset moves.

Stopping a Cam

Like other motion generators (jog, move, gear) active cams must be stopped by the various stop instructions, such as the Motion Axis Stop (MAS) on page 82 or

the Motion Group Stop (MGS) on page 196. Cam motion must also stop when the ControlLogix processor changes OS modes. The MAS instruction, in particular, must be able to specifically stop the camming process. This behavior should be identical to the MAS functionality that specifically stops a gearing process.

Merging from a Cam

Like other motion generators (jog, move, gear) active cams must also be compliant with motion merge functionality. Moves and Jogs, in particular, must be able to merge from active camming. This behavior should be identical to the merge functionality applied to a gearing process.

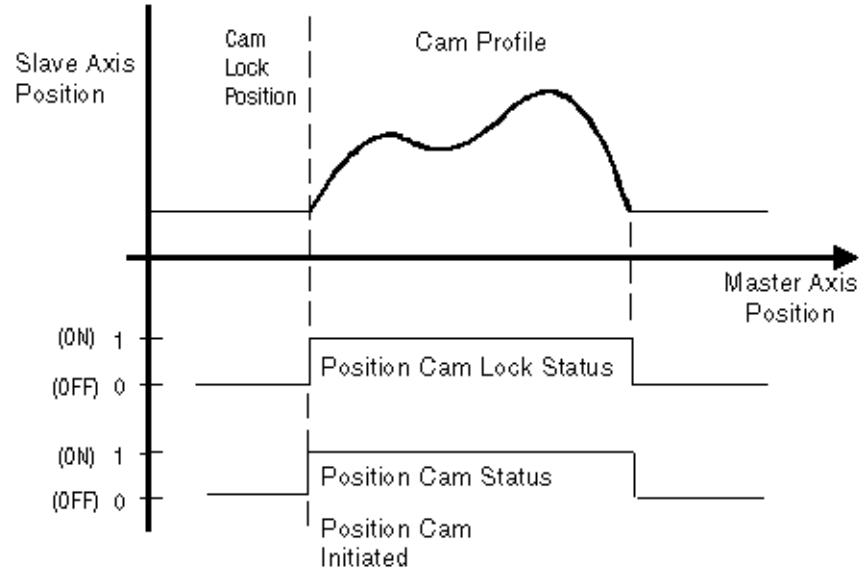
Handling Axis Faults

Sometimes it is necessary to respond to an axis fault condition without loosing synchronization between a master and slave axis that are locked in a cam relationship. With an active cam there are a couple ways to handle axis faults.

Create a virtual axis and cam everything to it and, if necessary, gear this virtual master axis to actual master axis of the machine. Set the various fault actions for all axes to Status Only. When an axis fault occurs (for example, a drive fault) an application program monitoring the axes fault status detects the fault and does a controlled stop of all active axes by stopping the virtual master axis. At the profiler level, everything is still fully synchronized. Use the following error on faulted axis to determine how far it is out of position. Reset the fault on the faulted axis, bring into position at a controlled speed by using the MAM instruction and the computed following error. Finally, start moving virtual master axis.

Same configuration as described previously but, in this case, when the slave axis faults the axis fault action disables the drive. This, of course, would terminate the active cam process on the slave axis. At this point, the application program should stop all other axes via the virtual master axis. Next, reposition the faulted axis by determining where the master is, and then calculating where the slave axis should be had the fault not occurred. Finally, do an immediate lock MAPC to resynchronize with the Cam Lock Position set to the calculated value.

Figure 16 - Position Cam Timing Diagram



Arithmetic Status Flags

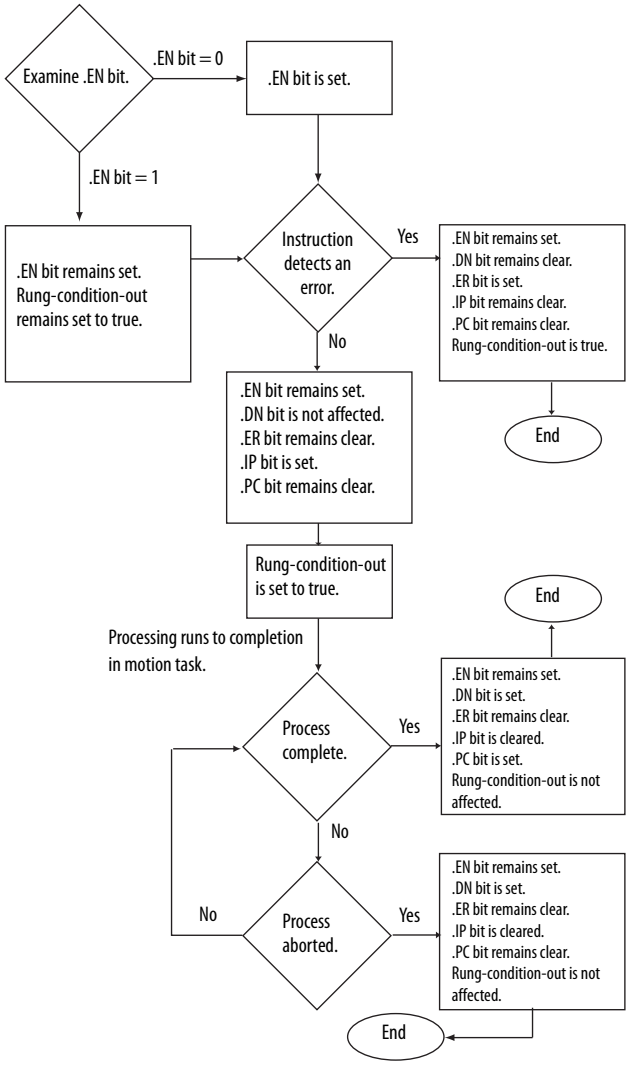
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAPC Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MAPC Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAPC Extended Error Codes

Use Extended Error Codes (EXERR) for more information about an error. Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions.

Table 64 - MAPC Extended Error Codes Description

If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
11	Varies	Slave or master axis is not configured.	Configure the axis.	
			1	Slave
			2	Master
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed
			4	
			5	Slave scaling value
11	Varies	Slave or master axis is not configured.	Configure the axis.	
			1	Slave
			2	Master
54	-1	The coordinate system has a Maximum Deceleration of 0.	Click the Properties for the coordinate system axis and set a Maximum Deceleration.	
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	If the Extended Error returns a positive number (0-n) it is referring to the offending axis in the coordinate system. Click the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. 1. Click the ellipsis button next to the offending axis to access the Axis Properties screen. 2. Click the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. 3. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Click the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.	

MAPC Changes to Status Bits

If the Execution Schedule is set to Immediate, execution of the MAPC instruction simply sets both the Position Cam Status and the Position Cam Lock Status bits to True.

Bit Name	State	Meaning
Position Cam Status	TRUE	Position Camming is Enabled
Position Cam Lock Status	TRUE	Slave Axis is Locked to the Master Axis according to the Cam Profile.
Position Cam Pending Status	Unchanged	

If the Execution Schedule is set to Forward or Reverse, execution of the MAPC instruction initially sets the Position Cam Status bit to True and the Position Cam Lock Status bits to False. Position Cam Lock Status transitions to True when the Execution Schedule condition is satisfied.

Bit Name	State	Meaning
Position Cam Status	TRUE	Position Camming is Enabled
Position Cam Lock Status	FALSE	Slave Axis is waiting for Master Axis to reach Lock Position.
Position Cam Pending Status	Unchanged	

If the Execution Schedule is set to Pending, execution of the MAPC instruction does not affect the current state of either the Position Cam Status or Position Lock Status bits. Position Cam Pending Status bit is set to True immediately and transitions to False when the pending cam becomes the active cam.

Bit Name	State	Meaning
Position Cam Status	Unchanged	
Position Cam Lock Status	Unchanged	
Position Cam Pending Status	True	Pending Position Cam

Once the MAPC is executing, the state of Position Cam Lock Status is determined by the setting of MasterDirection, and the direction the Master Axis is moving.

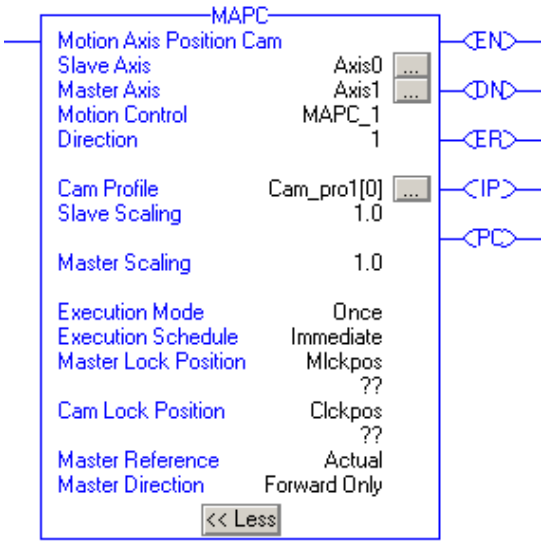
Bit Name	State	Meaning
Position Cam Status	True	Position Camming is Enabled
Position Cam Lock Status	True	Slave Axis is Locked to the Master Axis according to the Cam Profile.
	False	Slave Axis is waiting for the Master Axis to move in the direction specified by MasterDirection.
Position Cam Pending Status	Unchanged	

When ExecutionMode is set to Persistent, the Slave Axis will unlock whenever the Master Axis exceeds the Cam Profile range, and will lock when the Master Axis reenters the Cam Profile range.

Bit Name	State	Meaning
Position Cam Status	True	Position Camming is Enabled
Position Cam Lock Status	True	Slave Axis is Locked to the Master Axis according to the Cam Profile.
	False	Slave Axis is waiting for Master Axis to reenter Cam Profile range.
Position Cam Pending Status	Unchanged	

MAPC Example

Relay Ladder



Structured Text

```
MAPC(Axis0,Axis1,MAPC_1,1,Cam_pro1[0],1.0,1.0,Once,
immediate,Mlckpos,Clckpos,Actual,Forwardonly);
```

Motion Axis Time Cam (MATC)

The Motion Axis Time Cam (MATC) instruction provides electronic camming of an axis as a function of time, according to a specified time cam profile. The MATC instruction executes a time cam profile set up by a previous Motion Calculate Cam Profile (MCCP) instruction or, alternatively, by the Logix Designer Cam Profile Editor.

Time cams provide the capability of implementing complex motion profiles other than the built-in trapezoidal and S-curve motion profiles provided. No maximum velocity, acceleration, or deceleration limits are used in this instruction. The speed, acceleration, and deceleration of the slave axis are completely determined by the designated cam profile derived from the associated cam table.

ATTENTION: The maximum velocity, acceleration, or deceleration limits established during axis configuration do not apply to electronic camming.

Operands

The MATC instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

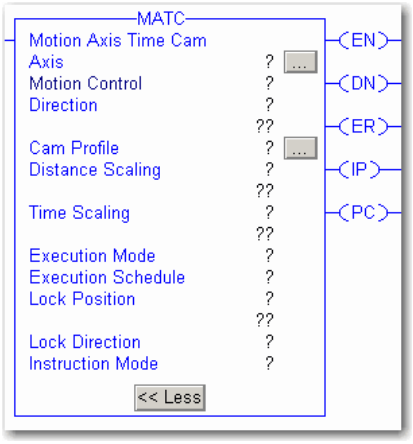


Table 65 - MATC Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	The name of the axis to which the cam profile is applied. Ellipsis launches Axis Properties dialog box.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.

Table 65 - MATC Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Direction	DINT	Immediate Tag	Relative direction of the slave axis to the master axis. Cams can be configured to add or subtract their incremental contribution to the axis command position. Control over this behavior is via the Direction parameter. 0 = Same The axis position values in the cam profile are added to the command position of the axis. When Same is selected or entered as the Direction for the MATC instruction, the axis position values computed from the cam profile are added to the command position of the axis. This is the most common operation, as the profile position values are used just as entered in the original cam table. That is, consecutive increasing profile values result in axis motion in the positive direction and vice-versa. 1 = Opposite The axis position values in the cam profile are subtracted from the command position of the axis creating axis motion in the other direction from that implied in the original cam table or relative to the current or previous camming direction. When Opposite is selected or entered as the Direction, the axis position values computed from the cam profile are subtracted from the command position of the axis. Thus, axis motion is in the opposite direction from that implied by the original cam table. That is, consecutive increasing profile values result in axis motion in the negative direction and vice-versa. 2 = Reverse The current or previous direction of the position cam is changed either from Same to Opposite or vice versa. When executed for the first time with Reverse selected, the control defaults the direction to Opposite. Changing the Camming Direction When Reverse is selected the current or previous direction of the time cam is changed from Same to Opposite or from Opposite to Same. For first time execution of a cam with Reverse selected, the control defaults the direction to Opposite. 3 = Unchanged This allows other cam parameters to be changed without altering the current or previous camming direction. When executed for the first time with Unchanged selected, the control defaults the direction to Same. When Unchanged is selected or entered as the Direction, other time cam parameters can be changed while preserving the current or previous camming direction (same or opposite). This is useful when the current direction is not known or not important. For first time execution of a cam with Unchanged selected, the control defaults the direction to Same.
Cam Profile	CAM_PROFILE	Array	Tag name of the calculated cam profile array. Only the zero array element ([0]) is allowed for the Cam Profile array. Ellipsis launches Cam Profile Editor.
Distance Scaling	REAL	Immediate Tag	Scales the total distance covered by the axis through the cam profile.
Time Scaling	REAL	Immediate Tag	Scales the time interval covered by the cam profile.
Execution Mode	DINT	Immediate	Determines how the cam motion behaves when the time moves beyond the end point of the cam profile. Execution Modes of Once or Continuous can be selected to determine how the cam motion behaves when the time moves beyond the end point of the profile defined by the original cam table. The options are: 0 = Once When the time cam execution time exceeds the time range in the cam profile, the MATC instruction completes, the axis motion stops, and the Time Cam Status bit is cleared. If Once is selected (default), the cam profile motion of the axis starts immediately. When the time cam execution time exceeds the time range defined by the cam profile, the MATC instruction completes, axis motion stops, and the Time Cam Status bit in the slave axis' Motion Status word is cleared. 1 = Continuous The cam profile motion is executed indefinitely. When Continuous mode is selected, the specified cam profile, starts immediately and is executed indefinitely. With continuous operation, time is `unwound' to the beginning of the cam profile when it moves beyond the end of the cam profile, causing the cam profile to repeat indefinitely. This feature is particularly useful in rotary applications where it is necessary that the time cam run continuously in a rotary or reciprocating fashion. To generate smooth continuous motion by using this technique, however, care must be taken in designing the cam points of the cam table to ensure that there are no position, velocity, or acceleration discontinuities between the start and end points of the calculated cam profile.

Table 65 - MATC Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Execution Schedule	DINT	Immediate	Selects the method used to execute the cam profile. Options are: 0 = Immediate – instruction is scheduled to execute immediately with no delay enabling the time camming process. 1 = Pending – lets you blend a new position cam execution after an in process position cam is finished.
Lock Position	REAL or TAG	Immediate Tag	Position on the Master Axis where a Slave axis should start after the move has been initiated on the Slave, only if the MATC instruction is specified in Master Driven mode, otherwise use a 0. Master Units
Lock Direction	REAL or TAG	Immediate	Specifies when the Master Lock Position should be used. Only used if the MATC instruction is specified in Time Driven mode. 0 = None, default 1= Immediate Forward Only 2 = Immediate Reverse Only 3 = Position Forward Only 4 = Position Reverse Only
Instruction Mode	DINT	Immediate	Specifies if the MATC instruction is in Time Drive or Master Driven mode.

Structured Text

MATC(Axis,MotionControl, Direction,CamProfile,
DistanceScaling,TimeScaling, ExecutionMode,
ExecutionSchedule,LockPosition, LockDirection, InstructionMode);

The operands are the same as those for the relay ladder MATC instruction. For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]).

Enter your selection for the operands that require you to select from available options.

Table 66 - MATC Available Operands

This Operand	Has These Options, That You Enter as Text	Or Enter as a Number
ExecutionMode	once continuous	0 1
ExecutionSchedule	immediate pending	0 1

MATC MOTION_INSTRUCTION Structure

Table 67 - MATC Bit Descriptions

Enumerations	Description
.EN (Enable) Bit 31	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN (Done) Bit 29	The done bit is set when the axis time cam instruction is successfully initiated.

Table 67 - MATC Bit Descriptions

Enumerations	Description
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if the axis is not configured.
.IP (In Process) Bit 26	The in process bit is set on positive rung transition and cleared when terminated by a stop command, merge, shutdown, or servo fault.
.PC (Process Complete) Bit 27	The Process Complete bit is cleared on positive rung transition and set in Once Execution Mode, when the time leaves the time range defined by the currently active cam profile.

Description

Use the MATC instruction for electronic camming of an axis according to a specified time cam profile.

The direction of axis motion relative to the cam profile is defined by a very flexible Direction input parameter. The camming Direction can be explicitly set as the Same or Opposite or set relative to the current camming direction as Reverse or Unchanged.

The cam profile can be configured via the Execution Schedule parameter to execute Immediately or Pending completion of a currently executing time cam profile. The cam profile can also be executed Once or Continuously by specifying the desired Execution Mode. Distance and Time Scaling functionality can be used to scale axis motion based on a standard cam profile without having to create a new cam table and calculate a new cam profile.

Linear and Cubic Interpolation

Time cams are fully interpolated. This means that if the current master time value does not correspond exactly with a point in the cam table associated with the cam profile, the slave axis position is determined by linear or cubic interpolation between the adjacent points. In this way, the smoothest possible slave motion is provided.

Each point in the Cam array that was used to generate the Cam Profile can be configured for linear or cubic interpolation.

Electronic camming remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic camming motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

Executing the Instruction

IMPORTANT	The MATC instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process (.IP) bit are set immediately. The In Process (.IP) bit remains set until the initiated Time Camming process is superseded by another MATC instruction, or terminated by a Motion Axis Stop command, Merge operation, or Servo Fault Action.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  For more information, see Structured Text Programming on page 359.

Specifying and Executing the Cam Profile

To execute a MATC instruction, a calculated Cam Profile data array tag must be specified. Cam Profile array tags can be created by the Logix Designer tag editor or the MATC instruction by using the built-in Cam Profile Editor, or by executing an Motion Calculate Cam Profile (MCCP) instruction on an existing Cam array.

The data within the Cam Profile array can be modified at compile time by using the Cam Profile Editor, or at run-time with the Motion Calculate Cam Profile (MCCP) instruction. In the case of run-time changes, a Cam array must be created in order to use the MCCP instruction.

The Status member of this Cam Profile Array is used to indicate that the Cam Profile array element has been calculated. If execution of a camming instruction is attempted with any uncalculated elements in a cam profile, the instruction errors. The type parameter determines the type of interpolation applied between this cam array element and the next cam element.

The Status member of the first element in the cam profile array is special and used for data integrity checks. For this reason, the MATC must always specify the cam profile with the starting index set to 0. This first cam profile element Status member can have the following values.

Table 68 - MATC Status Member Value Descriptions

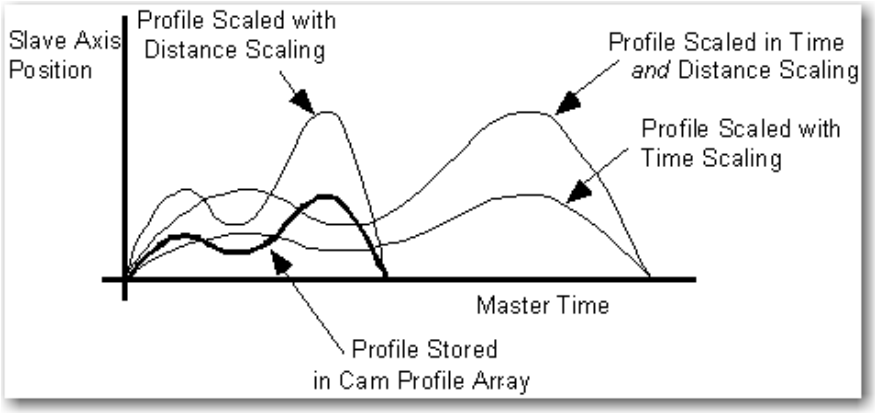
Status Value	Description
0	Cam profile element has not been calculated.
1	Cam profile element is being calculated.
2	Cam profile element has been calculated.
n	Cam profile element has been calculated and is currently being used by (n-2) MAPC or MATC instructions.

Before starting a cam on a specified axis, the MATC instructions checks if the cam profile array has been calculated by checking the value of the first cam profile element's Status member. If Status is 0 or 1 then the cam profile has not been calculated yet and the MATC instruction errors. If the cam profile array has been completely calculated (Status > 1), the instruction then increments the Status member indicating that it is in use by this axis.

When the cam completes, or terminates, the Status member of the first cam profile array element is decremented to maintain track of the number of cams actively by using the associated cam profile.

Scaling Time Cams

A time cam profile can be scaled in both time and distance when it is executed. This scaling is useful to allow the stored profile to be used only for the form of the motion with the scaling used to define the time or distance over which the profile is executed, as shown in the following figure.


Slave Axis Position
Profile Scaled with Distance Scaling
Profile Scaled in Time and Distance Scaling
Profile Scaled with Time Scaling
Master Time
Profile Stored in Cam Profile Array

When a cam profile array is specified by an MATC instruction, the master coordinate values defined by the cam profile array take on the time units (seconds) and the slave values take on the units of the slave axis. By contrast, the Time and Distance Scaling parameters are 'unitless' values that are simply used as multipliers to the cam profile.

By default, both the Time and Distance Scaling parameters are set to 1. To scale a time cam profile, enter a Time Scaling or Distance Scaling value other than 1.

Increasing the Time Scaling value of a cam profile decreases the velocities and accelerations of the profile, while increasing the Distance Scaling value increases the velocities and accelerations of the profile. To maintain the velocities and accelerations of the scaled profile approximately equal to those of the unscaled profile, the Time Scaling and Distance Scaling values should be equal.

For example, if the Distance Scaling value of a profile is 2, the Time Scaling value should also be 2 to maintain approximately equal velocities and accelerations during execution of the scaled time cam.

IMPORTANT	Decreasing the Time Scaling value or increasing the Distance Scaling of a time cam increases the required velocities and accelerations of the profile. This can cause a motion fault if the capabilities of the drive system are exceeded.

Execution Schedule

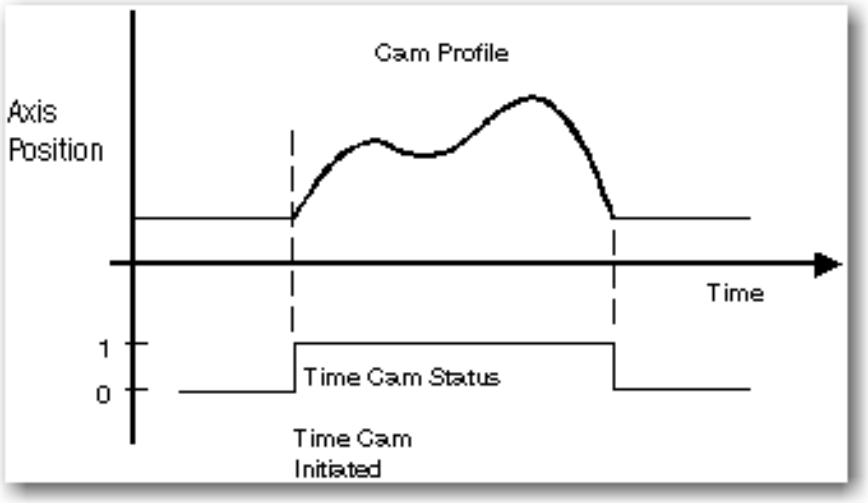
Control over the MATC instruction's execution schedule is via the Execution Schedule parameter.

Immediate Execution

By default, the MATC instruction is scheduled to execute immediately by virtue of the fact that the default setting of the Execution Schedule parameter is Immediate. In this case, there is no delay to the enabling of the time camming process.

As illustrated in Figure 17, when the MATC instruction is executed, the camming process is initiated on the specified axis and the Time Cam Status bit in the axis' Motion Status word is set. If the Execution Schedule parameter is set to Immediate, the axis is immediately locked to the time master coordinate according to the specified Cam Profile.

Figure 17 - Camming Process Initiated



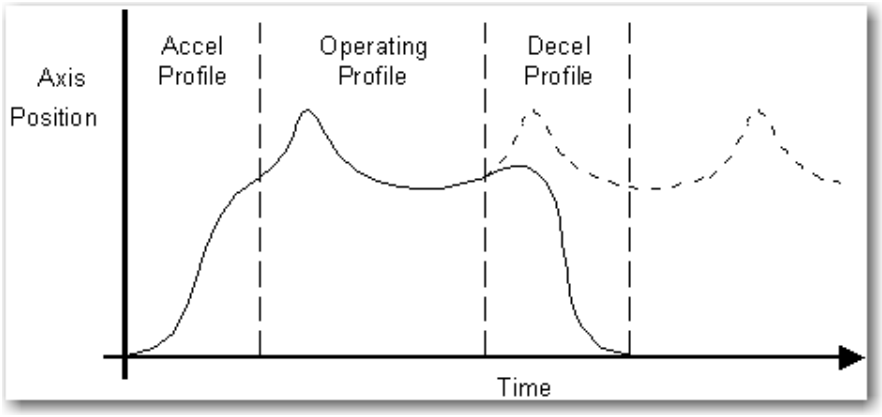
If an MATC instruction is executed on an axis that is already actively time camming, an Illegal Dynamic Change error is generated (error code 23). The only exception for this is if the Execution Schedule is specified as 'pending'.

Pending Cam Execution

The transition from one cam profile to another occurs when the operational cam profile reaches the last point in its table. At this point the pending cam profile table goes into operation.

Alternatively, the MATC instruction's execution can, in effect, be deferred pending completion of a currently executing time cam. An Execution Schedule selection of Pending can thus be used to seamlessly blend two time cam profiles together without stopping motion. As with the continuous mode of operation, the blending points should not have large value differences. Large differences can result in dramatic acceleration and deceleration changes. Blending points are the start and end points for the continuous cam profile. In other words, the points are the last point of the operating cam profile and first point of pending cam profile.

The Pending execution feature is particularly useful in applications when the axis must be accelerated up to speed by using a specific velocity profile. When this acceleration profile is done, it must be smoothly blended into a cam profile which is typically executed continuously. To stop the axis, the operating profile can be smoothly blended into a deceleration profile such that the axis stops at a known location as shown in the following figure.



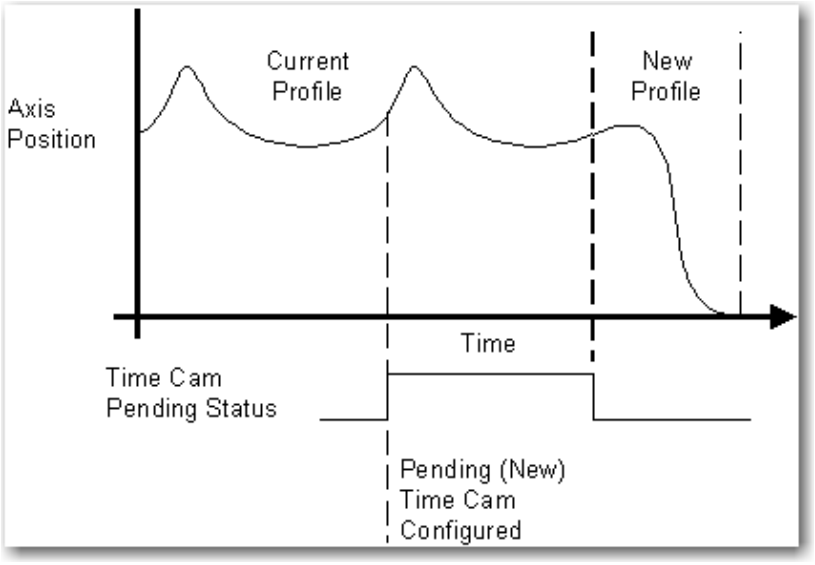
By executing the time cam profile as a Pending cam profile while the current profile is still executing, the appropriate cam profile parameters are set up ahead of time. This makes the transition from the current profile to the pending profile seamless – synchronization between the master time and slave axes position is maintained. To ensure smooth motion across the transition, however, the profiles must be designed such that no position, velocity, or acceleration discontinuities exist between the end of the current profile and the start of the new one. This is done by using the Logix Designer Cam Profile Editor.

Once a pending time cam instruction has been executed, the new cam profile takes effect automatically (and becomes the current profile) when cam time passes through the end of the current profile. If the current cam is configured to execute once, the new profile is initiated at the completion of the pass through the current cam profile and the PC bit of the currently active MATC instruction is set. If the current cam is configured to execute continuously, the new profile is initiated at the completion of the current pass through the current cam profile and the IP bit of the currently active MATC instruction is cleared. The motion controller keeps track of time and the axis positions relative to the first profile at the time of the change and uses this information to maintain synchronization between the profiles.

If the Execution Schedule of an MATC instruction is set to Immediate and a time cam profile is currently in process, the MATC instruction generates an Illegal Dynamic Change error.

If an Execution Schedule of Pending is selected without a corresponding time cam profile in progress, the MATC instruction executes but no camming motion occurs until another MATC instruction with a non-pending Execution Schedule is initiated. This allows pending cam profiles to be preloaded prior to executing the initial cam. This method addresses cases where immediate cams would finish before the pending cam could be reliably loaded.

After a Pending time cam has been configured, the Time Cam Pending Status bit of the Motion Status word for the specified axis is set to 1 (true). When the pending (new) profile is initiated and becomes the current profile, Time Cam Pending Status bit is immediately cleared as shown in the following figure.

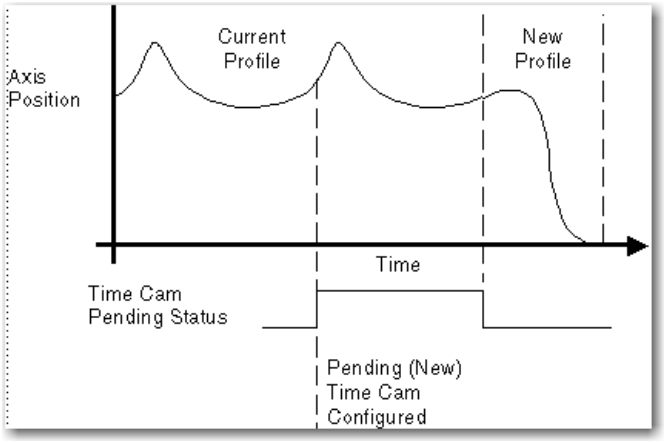


Stopping a Cam

Like other motion generators (jog, move, gear) active cams must be stopped by the various stop instructions, such as the Motion Axis Stop (MAS) on page 82 or the Motion Group Stop (MGS) on page 196. Cam motion must also stop when the ControlLogix processor changes OS modes. The MAS instruction, in particular, must be able to specifically stop the camming process. This behavior should be identical to the MAS functionality that specifically stops a gearing process.

Merging from a Cam

Like other motion generators (jog, move, gear, and so forth) active cams must also be compliant with motion merge functionality. Moves and Jogs, in particular, must be able to merge from active camming. This behavior should be identical to the merge functionality applied to a gearing process.



Arithmetic Status Flags

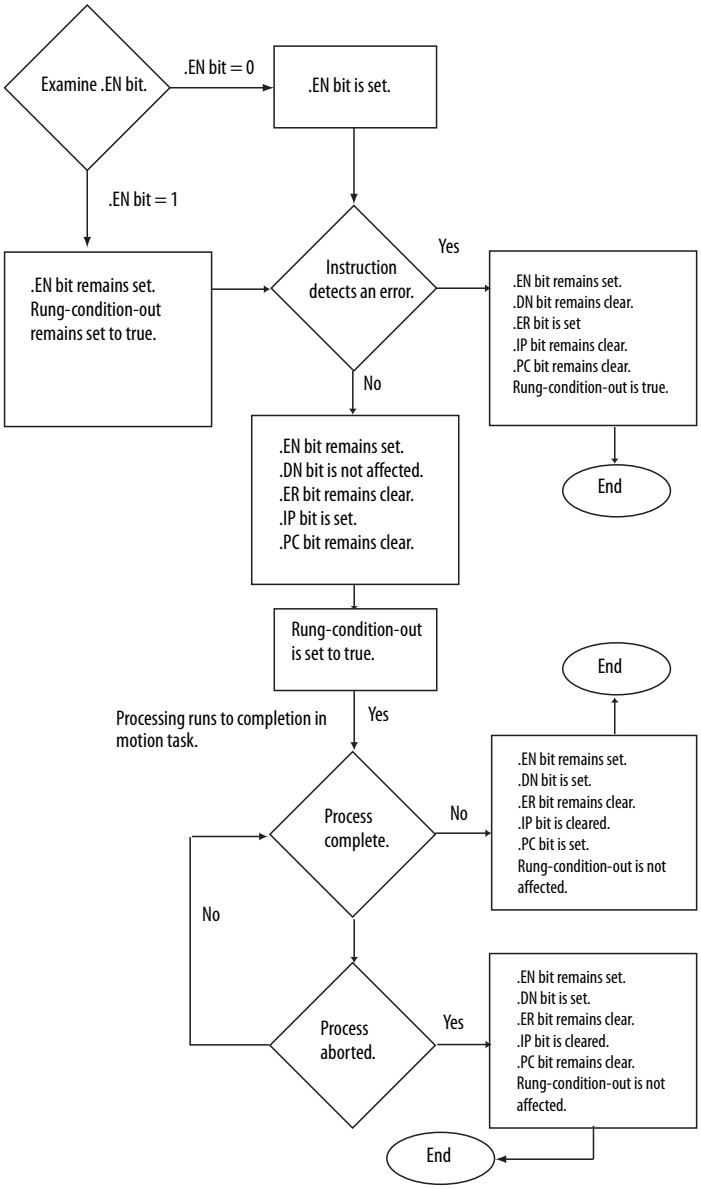
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MATC Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



MATC Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MATC Extended Error Codes

Extended Error Codes provide additional instruction-specific information for the Error Codes that are generic to many instructions. Extended error codes for the PARAMETER_OUT_OF_RANGE(13) error code lists a number that refers to the number of the operands, as they are listed in the faceplate, from top to bottom, with the first operand being counted as zero. Therefore, for the MATC instruction, an extended error code of 5 would refer to the Time Scaling

operand's value. You would then have to check your value against the accepted range of values for the instruction.

MATC Changes to Status Bits

If the Execution Schedule is set to Immediate, execution of the MATC instruction simply sets the Time Cam Status bit to True.

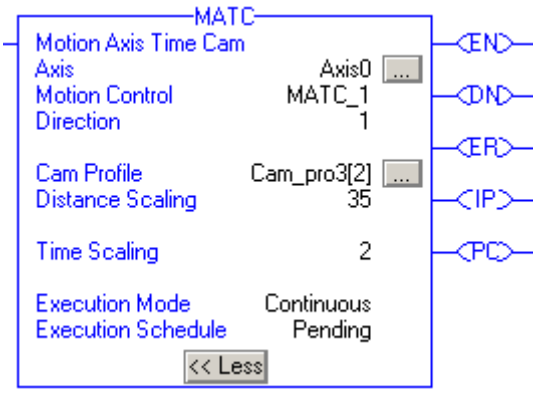
Bit Name	State	Meaning
TimeCamStatus	TRUE	Time Camming is Enabled
TimeCamPendingStatus	FALSE	No pending Time Cam

If the Execution Schedule is set to Pending, execution of the MATC instruction does not affect the current state of the Time Cam Status bits. Time Cam Pending Status bit is set to True immediately and transitions to False when the pending cam becomes the active cam.

Bit Name	State	Meaning
TimeCamStatus	N/A	Time Camming is Enabled
TimeCamPendingStatus	TRUE	Pending Time Cam

MATC Example

Relay Ladder



Structured Text

```
MATC(Axis0,MATC_1,1,Cam_pro3[2],35,2,Continuous,
Pending);
```

Motion Calculate Slave Values (MCSV)

Use the Motion Calculate Slave Values (MCSV) instruction to calculate the slave value, the slope value, and the derivative of the slope for a given cam profile and master value.

Operands

The MCSV instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

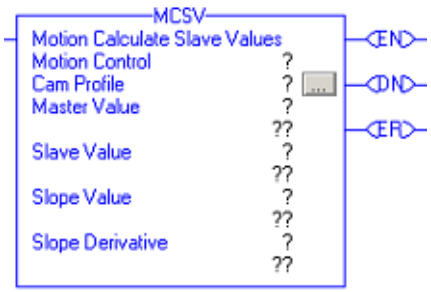


Table 69 - MCSV Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Cam Profile	CAM_PROFILE	Array Tag	An array of elements with the array index set to 0. It defines the cam profile used in calculating the slave values.
Master Value	SINT, INT, DINT, or REAL	Immediate Tag	The exact value along the master axis of the cam profile that is used in calculating the slave values.
Slave Value	REAL	Tag	The value along the slave axis of the cam profile with the master at the specified master value.
Slope Value	REAL	Tag	The first derivative of the value along the slave axis of the cam profile with the master at the specified master value.
Slope Derivative	REAL	Tag	The second derivative of the value along the slave axis of the cam profile with the master at the specified master value.

Structured Text

MCSV(MotionControl,CamProfile, MasterValue,SlaveValue, SlopeValue,SlopeDerivative)

The operands are the same as those for the relay ladder MCSV instruction.

MCSV MOTION_Instruction Structure

The following control bits are affected by the MCSV instruction.

Table 70 - MCSV Bit Descriptions

Enumerations	Description
.EN (Enable) Bit 31	The Enable Bit sets when the rung transitions from false to true. It resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done Bit sets when the slave values have been calculated successfully. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error Bit sets when the slave values have not been calculated successfully. It resets when the rung transitions from false to true.

Description

The MCSV instruction determines the slave value, the slope value, and the derivative of the slope for a given cam profile and master value. As an extension to the position and time camming functionality it supplies the values essential for the recovery from faults during camming operations.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it executes only on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MCSV Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MCSV Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. Extended Error Codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MCSV instruction, an extended error code of 2 would refer to the Master Value operand's value. You would then have to check your value with the accepted range of values for the instruction.

MCSV Changes to Status Bits

The MCSV instruction does not make any changes to the status bits.

Motion Group Instructions
(MGS, MGSD, MGSR, MGSP)

ATTENTION: Tags used for the motion control attribute of instructions should only be used once. Reuse of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

Group Control Instructions include all motion instructions that operate on all the axes in the specified group. Instructions that can be applied to groups include position strobe, shutdown control, and stopping instructions. Only one group is supported per Logix controller.

These are the motion group instructions.

Table 71 - Choosing Motion Group Instructions

If You Want To	Use This Instruction	Page	Languages
Initiate a stop of motion on a group of axes.	Motion Group Stop (MGS)	196	Relay ladder Structured text
Force all axes in a group into the shutdown operating state.	Motion Group Shutdown (MGSD)	202	
Transition a group of axes from the shutdown operating state to the axis ready operating state.	Motion Group Shutdown Reset (MGSR)	206	
Latch the current command and actual position of all axes in a group.	Motion Group Strobe Position (MGSP)	210	

Motion Group Stop (MGS)

The Motion Group Stop (MGS) instruction initiates a stop of all motion in progress on all axes in the specified group by a method configured individually for each axis or as a group via the Stop Mode of the MGS instruction. If the MGS Stop Mode is specified as Programmed, each axis in the group is stopped according to the configured Programmed Stop Mode axis attribute. This is the same stopping mechanism that is employed by the Logix Operating System when there is a Logix controller state change. This Programmed Stop Mode attribute currently provides five different methods of stopping an axis:

- Fast Stop
- Fast Disable
- Hard Disable
- Fast Shutdown
- Hard Shutdown

Alternatively, an explicit Stop Mode can be selected by using the MGS instruction. If a Stop Mode of Fast Disable is selected, all axes in the group stop with Fast Disable behavior. When the motion of all the axes in the group has been brought to a stop, the Process Complete (PC) bit is set in the control structure.

Operands

The MGS instruction supports the following operands:

- Relay Ladder
- Structured Text

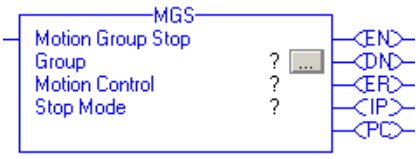


Table 72 - MGS Relay Ladder Descriptions

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Stop Mode	UDINT	Immediate	Controls how the axes in the group are stopped. Select one of the following methods: 0 = Programmed - each axis is stopped according to how the individual axis has been configured. 1 = Fast Stop - each axis in the group is decelerated at the Maximum Deceleration rate and the stopped axis is left in the Servo Active state. 2 = Fast Disable - each axis in the group is decelerated at the Maximum Deceleration rate and the stopped axis is placed in the Axis Ready state.

Structured Text

MGS(Group,MotionControl,StopMode);

The operands are the same as those for the relay ladder MGS instruction. Enter your selection for the operands that require you to select from available options.

Table 73 - MGS Structured Text

This Operand	Has These Options That You	
	Enter as Text	Or Enter as a Number
StopMode	programmed faststop fastdisable	0 1 2

MOTION_INSTRUCTION Structure

Table 74 - MGS MOTION_INSTRUCTION Structure Description

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the group Programmed Stop has been successfully initiated for all axes in the group.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured group.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the Motion Group Stop is complete.
.PC (Process Complete) Bit 27	It is set after all the axes in group have been successfully brought to a stop according to each axis' Programmed Stop Mode configuration.

Description

With the Stop Mode parameter set for Programmed, the MGS instruction brings motion for all of the axes in the specified group to a stop according to the configured Programmed Stop Mode for each axis. If the axis has both single axis motion moves and coordinated moves occurring, the MGS stops the single axis motion by using the axis' maximum deceleration rate and stops the coordinated axes motion by using the coordinate system's maximum deceleration rate. A trapezoidal profile is always used for the deceleration regardless of the programmed profile type.

The MGS instruction initiates the same programmed stopping action that is automatically applied when the processor's operating system changes operating mode (for example, Run Mode to Program Mode). This is particularly useful in designing custom motion fault handlers.

If the MGS Stop Mode parameter is set to Fast Stop, each axis in the group is forced to perform a Fast Stop process, regardless of the configured Programmed

Stop Mode. Each axis in the group is decelerated at the Maximum Deceleration rate and, once stopped, the axis is left in the Servo Active state.

If the MGS Stop Mode parameter is set to Fast Disable, each axis in the group is forced to perform a Fast Disable process, regardless of the configured Programmed Stop Mode. Each axis in the group is decelerated at the Maximum Deceleration rate and, once stopped, placed into the Axis Ready (servo inactive and drive disabled) state.

The MGS instruction currently supports five Programmed Stop Action modes:

- Fast Stop
- Fast Disable
- Hard Disable
- Fast Shutdown
- Hard Shutdown

Each axis can be configured to use any of these five stop modes. The following is a description of the effect of each these five stopping modes as they apply to an individual axis in the specified group.

Table 75 - MGS Programmed Stop Action Modes Descriptions

Mode	Description
Fast Stop	For an axis configured for a Fast Stop the MGS instruction initiates a controlled stop much like that initiated by an MAS instruction. In this case the Motion Group Stop (MGS) instruction brings the axis motion to a controlled stop without disabling the axis servo loop. It is useful when a fast decelerated stop the axis is desired with servo control retained. The MGS instruction uses the configured Maximum Deceleration of the axis to stop only the single axis motion. The coordinated move portion of the axis uses the coordinated system configured Maximum Deceleration to stop the axis. When a Fast Stop is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis decelerates to a stop by using its ramp deceleration.
Fast Disable	For an axis configured for a Fast Disable the MGS instruction initiates a controlled stop much like that initiated by an MAS instruction with the exception that the drive is disabled when the axis comes to a stop. Use MGS when a fast decelerated stop the axis is desired before the drive is disabled. The MGS instruction uses the configured Maximum Deceleration of the axis to stop only the single axis motion. The coordinated move portion of the axis uses the coordinated system configured Maximum Deceleration to stop the axis. Integrated Motion on the Ethernet/IP network When a Fast Disable is issued and a Direct Velocity command is issued via the MDS instruction, the CIP axis is disabled when all planned motion is stopped. However, in this case, the drive device continues coasting or stopping according to its Stop Action selection.
Hard Disable	For an axis configured for a Hard Disable the MGS instruction initiates the equivalent of an MSF instruction to the axis. This action immediately turns the appropriate axis drive output off, and disables the servo loop. Depending on the drive configuration, this can result in the axis coasting to a stop but offers the quickest disconnect of drive output power. When a Hard Disable is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis is immediately disabled.

Table 75 - MGS Programmed Stop Action Modes Descriptions (Continued)

Mode	Description
Fast Shutdown	For an axis configured for a Fast Shutdown, the MGS instruction initiates a Fast Stop and then applies the equivalent of a Motion Axis Shutdown (MASD) instruction to the axis. This action turns the appropriate axis driver output OFF, disables the servo loop, opens any associated motion module's OK contacts, and places the axis into the Shutdown state. When a Fast Shutdown is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis decelerates to a stop by using its ramp deceleration is immediately shutdown.
Hard Shutdown	For an axis configured for a Hard Shutdown the MGS instruction initiates the equivalent of an Motion Axis Shutdown (MASD) instruction to the axis. This action turns the appropriate axis drive output OFF, disables the servo loop, opens any associated motion module OK contacts, and places the axis into the Shutdown state. Depending on the drive configuration, this can result in the axis coasting to a stop but offers the quickest disconnect of Drive power via the OK contacts. To successfully execute a MGS instruction, the targeted group must be configured. When a Hard Shutdown is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis is immediately shutdown. The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In addition to the ways the various stopping modes affect the Motion Drive Start (MDS) instruction were described in the previous table, all these modes also clear the MDS In Process (.IP) bit and clear the DirectVelocityControlStatus bit in the Motion Status attribute.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

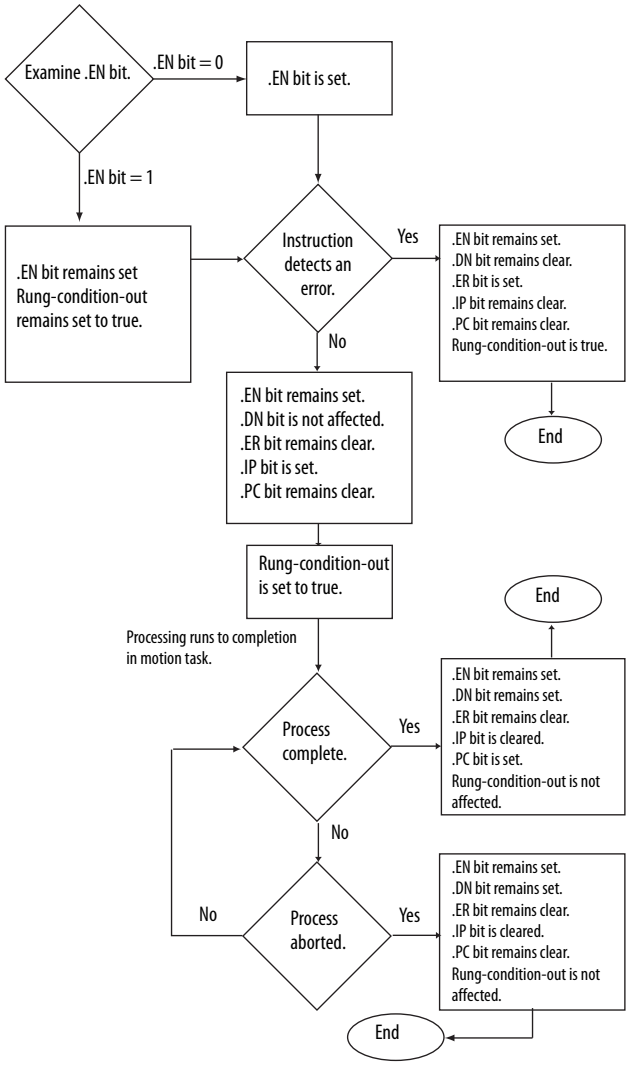
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MGS Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MGS Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MGS Changes to Status Bits

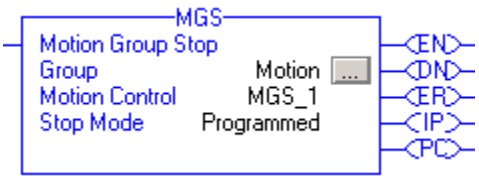
Table 76 - MGS Changes to Status Bits for the MSG Instruction

If the Stop Type is	Then		
NOT All	The instruction clears the Motion Status bit for the motion process that you stopped.		
All	The instruction clears all Motion Status bits.		
	Bit	**Status**	**Meaning**
	MoveStatus	FALSE	Axis is not Moving
	JogStatus	FALSE	Axis is not Jogging
	GearingStatus	FALSE	Axis is not Gearing
	HomingStatus	FALSE	Axis is not Homing
	StoppingStatus	TRUE	Axis is Stopping
	PositionCamStatus	FALSE	Axis is not Position Camming
	TimeCamStatus	FALSE	Axis is not Time Camming
	PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.
	TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
	GearingLockStatus	FALSE	Axis is not in a Gear Locked condition
	PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition
	DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
	DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

MGS Example

When the input conditions are true, the controller stops motion on all axes in *group1*. After the controller stops all motion, the axes are inhibited.

Relay Ladder



Structured Text

```
MGS(Motion,MSG_1,Programmed);
```

Motion Group Shutdown (MGSD)

Use the Motion Group Shutdown (MGSD) instruction to force all axes in the designated group into a Shutdown state. The Shutdown state of an axis is Servo Off, drive output is deactivated, and the motion module's OK solid-state relay contacts, if applicable, are opened. The group of axes remains in the Shutdown state until either Group Shutdown Reset is executed or each axis is individually reset via the Motion Axis Shutdown (MASD) instruction.

Operands

The MGSD instruction supports the following operands:

- Relay Ladder
- Structured Ladder

Relay Ladder

Table 77 - MGSD Relay Ladder Descriptions

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MGSD(Group,MotionControl);

The operands are the same as those for the relay ladder MGSD instruction.

MOTION_INSTRUCTION Structure

Table 78 - MGSD MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN (Done) Bit 29	The done bit indicates when the instruction sets the group of axes to the shutdown operating state.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.

Description

The MGSD instruction turns drive output off, disables the servo loops of all axes in the specified group, and opens any associated OK contacts for all applicable motion modules in the group. This action places all group axes into the Shutdown state. The MGSD instruction takes only one parameter; simply select or enter the desired group to shutdown.

Another action initiated by the MGSD instruction is the clearing of all motion processes in progress and a clearing of all the motion status bits. Associated with this action, the command also clears all motion instruction.IP bits that can currently be set for each axis in the group.

The MGSD instruction forces the targeted group of axes into the Shutdown state. One of the unique characteristics of the Shutdown state is that the OK solid state relay contact for all of the group's motion modules Open. This feature can be used to open up the E-Stop string(s) that control main power to the various drive systems.

Another characteristic of the Shutdown state is that any instruction that initiates axis motion for an axis within the group is blocked from execution. Attempts to do so results in an execution error. Only by executing one of the Shutdown Reset instructions can motion then be successfully initiated.

Executing the Instruction

To successfully execute a MGSD instruction, the targeted group must be created and configured.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

Additionally, the MGSD instruction supports canceling the Motion Drive Start (MDS) instruction. This includes clearing the MDS In Process (.IP) bit, and clearing the DirectVelocityControlStatus bit and the DirectTorqueControlStatus bit in the Motion Status attribute.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

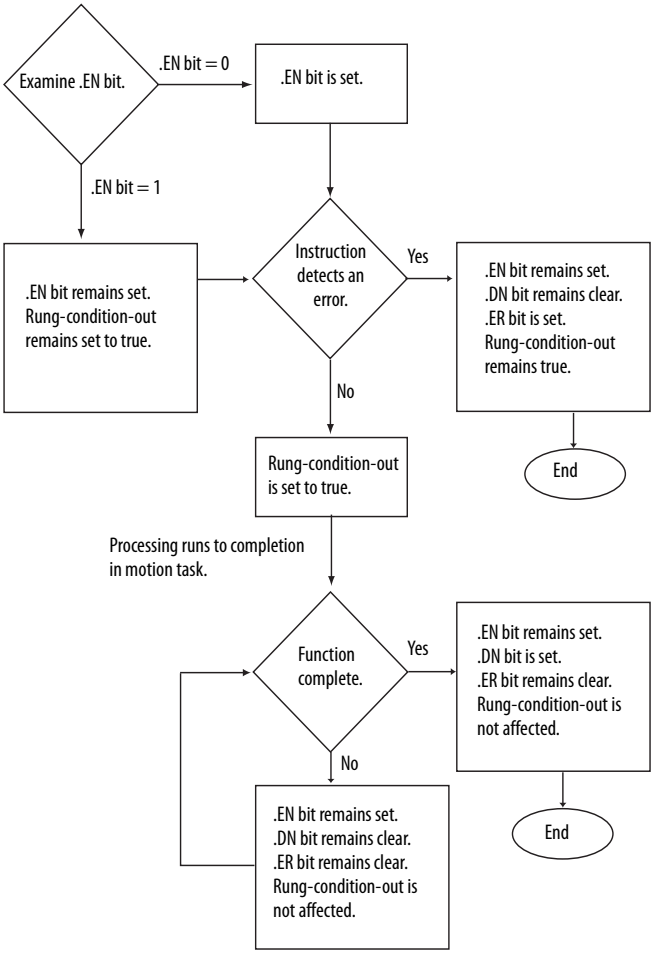
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MGSD Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, and .ER bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MGSD Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MGSD Changes to Status Bits

Table 79 - Changes to Status Bits for the MGSD Instruction

Bit Name	State	Definition
ServoActionStatus	FALSE	Axis is in Axis Ready state with the servo loop inactive.
DriveEnableStatus	FALSE	Axis Drive Enable output is inactive.
ShutdownStatus	TRUE	Axis is in Shutdown state.
AccelStatus	FALSE	Axis is not Accelerating
DecelStatus	FALSE	Axis is not Decelerating
GearingLockStatus	FALSE	Axis is not locked.
JogStatus	FALSE	Axis is not Jogging
MoveStatus	FALSE	Axis is not Moving
GearingStatus	FALSE	Axis is not Gearing
HomingStatus	FALSE	Axis is not Homing
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

MGSD Example

When the input conditions are true, the controller forces all axes in *group1* into a shutdown operating state.

Relay Ladder

Structured Text

```
MGSD(Motion,MGSD_2);
```

Motion Group Shutdown Reset (MGSR)

Use the Motion Group Shutdown Reset (MGSR) instruction to transition a group of axes from the shutdown operating state to the axis ready operating state. As a result of this command, all faults associated with the axes in the group are cleared and any OK relay contacts of motion modules associated with the specified group are closed.

Operands

The MGSR instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 80 - MGSR Relay Ladder Descriptions

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MGSR(Group,MotionControl);

The operands are the same as those for the relay ladder MGSR instruction.

MOTION_INSTRUCTION Structure

Table 81 - MGSR MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN (Done) Bit 29	The done bit indicates when the instruction resets the group of axes from the shutdown operating state.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.

Description

The MGSR instruction takes all the axes in the specified group out of the Shutdown state by clearing all axis faults and closing any associated OK solid-state relay contacts for the motion modules within the group. This action places all axes within the motion group in the Axis Ready state.

Just as the Motion Group Shutdown (MGSD) instruction forces all the axes in the targeted group into the Shutdown state. The MGSR instruction takes all the axis in the specified group out of the Shutdown state and into the Axis Ready state. One of the unique characteristics of the Shutdown state is that, if supported, the OK solid state relay contact for each of the group's motion modules is Open. Hence, the result of an MGSR instruction applied to a group of motion modules is that all motion module OK relay contacts close. This feature can be used to close the E-Stop strings that control main power to the various drive systems and permits the customer to reapply power to the drives.

Executing the Instruction

To successfully execute a MGSR instruction, the targeted group must be configured.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

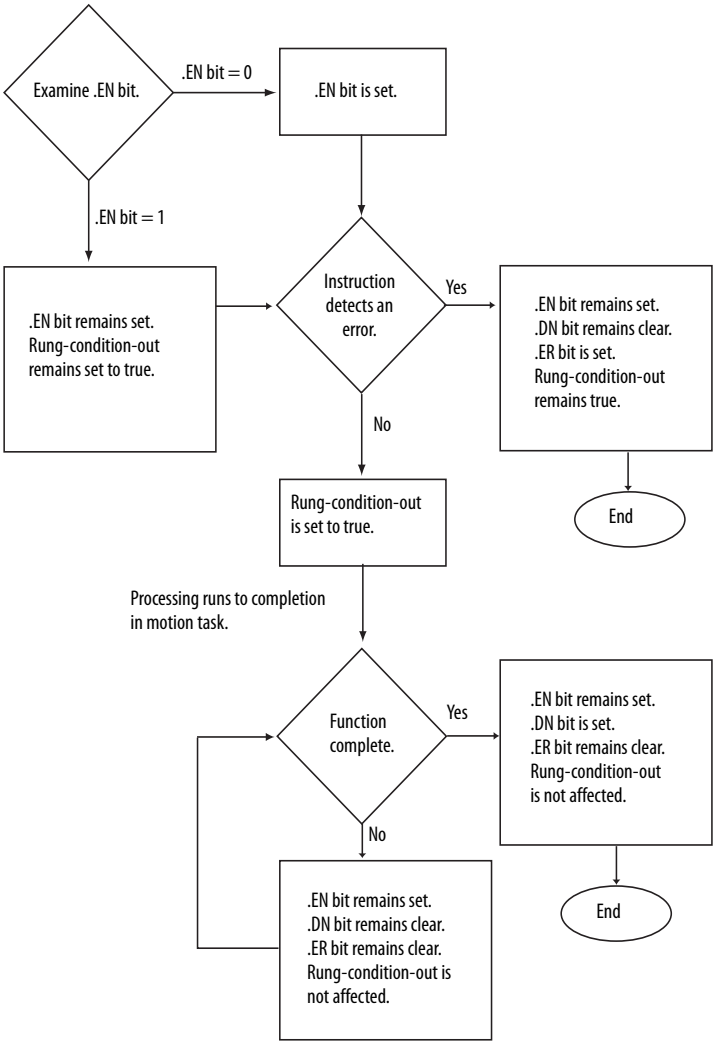
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MGSR Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, and .ER bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MGSR Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MGSR Changes to Status Bits

Bit Name	State	Definition
ServoActionStatus	FALSE	Axis is in Axis Ready state with the servo loop inactive.
DriveEnableStatus	FALSE	Axis Drive Enable output is inactive.
ShutdownStatus	FALSE	Axis is NOT in Shutdown state.

MGSR Example

When the input conditions are true, the controller transitions all axes in *group1* from the shutdown operating state to the axis ready operating state.

Relay Ladder

Structured Text

```
MGSR(Motion,MGSR_3);
```

Motion Group Strobe Position (MGSP)

Use the Motion Group Strobe Position (MGSP) instruction to latch the current Command and Actual Position of all axes in the specified group at a single point in time. The latched positions are available in the StrobeActualPosition and StrobeCommandPosition parameters in the Motion Axis Object for each axis configured in the group.

Operands

The MGSP instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 82 - MGSP Relay Ladder Descriptions

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MGSP(Group,MotionControl);

The operands are the same as those for the relay ladder MGSP instruction.

MOTION_INSTRUCTION Structure

Table 83 - MGSP MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the group of axes have been successfully set to Shutdown state.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured group.

Description

The MGSP instruction synchronously latches all command and actual position values of all axes in the specified group at the time of execution. The MGSP instruction takes only one parameter; simply select or enter the desired axis to strobe.

If the targeted group does not appear in the list of available groups, the group has not been configured for operation. Use the Tag Editor to create and configure a new groups.

The MGSP instruction can be used at any time to capture a complete set of command and actual position information for all axes in the specified group. This operation is often required as a precursor to computations involving position values of different axes within the group.

Executing the Instruction

To successfully execute a MGSP instruction, the targeted group must be configured.

IMPORTANT	The MGSP instruction execution completes in a single scan, setting the Done .DN bit immediately.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

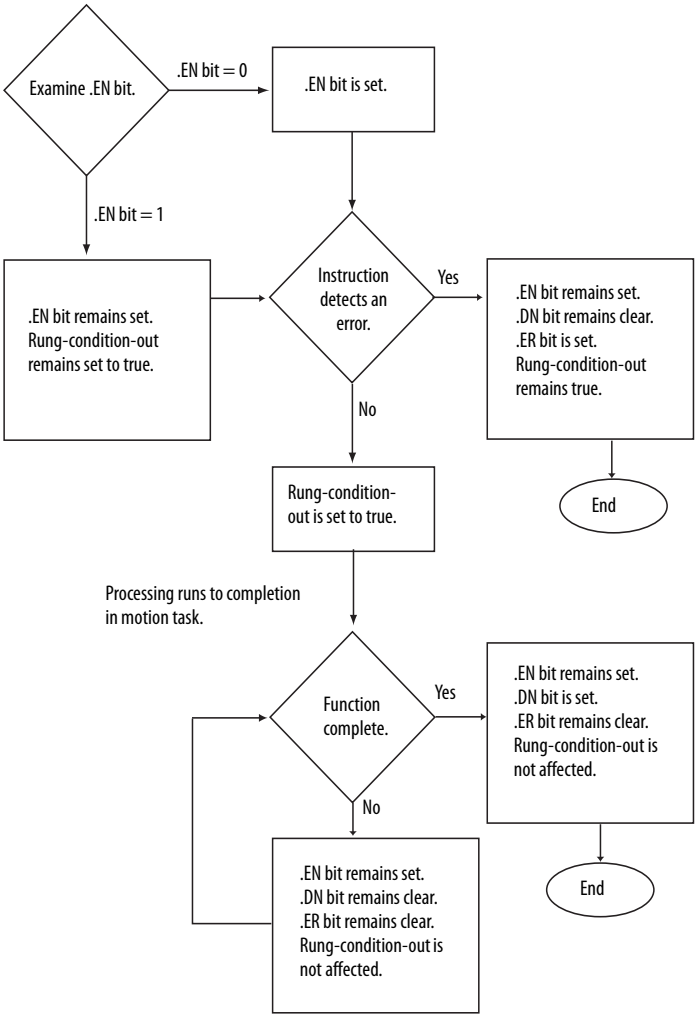
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MGSP Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, and .ER bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MGSP Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MGSP Changes to Status Bits

The MGSP instruction does not make any changes to the status bits.

MGSP Example

When the input conditions are true, the controller latches the current command and the actual position of all axes in *group1*.

Relay Ladder

Structured Text

```
MGSP(Motion,MGSP_2);
```

Notes:

Motion Event Instructions
(MAW, MDW, MAR, MDR, MAOC, MDOC)

ATTENTION: Tags used for the motion control attribute of instructions should only be used once. Reuse of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

These are the motion event instructions.

Table 84 - Choosing a Motion Event Instruction

If You Want To	Use This Instruction	Page	Languages
Arm watch-position event-checking for an axis.	Motion Arm Watch (MAW)	216	Relay ladder Structured text
Disarm watch-position event-checking for an axis.	Motion Disarm Watch (MDW)	223	
Arm servo-module registration-event checking for an axis.	Motion Arm Registration (MAR)	227	
Disarm servo-module registration-event checking for an axis.	Motion Disarm Registration (MDR)	236	
Arm an Output Cam	Motion Arm Output Cam (MAOC)	239	
Disarm an Output Cam	Motion Disarm Output Cam (MDOC)	256	

Motion Arm Watch (MAW)

Use the Motion Arm Watch (MAW) instruction to arm motion module watch position event checking for the specified axis. When this instruction is called, a watch position event is enabled by using the watch Position for the Axis and specified Forward or Reverse event condition. After the arming is complete the Actual Position for the Axis is monitored against the Watch Position and when the specified watch event condition is met, the Event (PC) bit is set, and the Watch Event Status bit in the Axis data structure is set.

Operands

The MAW instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

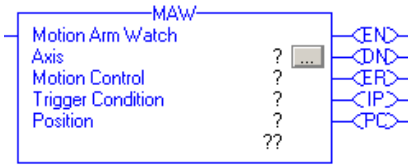


Table 85 - MAW Relay Ladder Operand Description

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Trigger condition	BOOLEAN	Immediate	Select the watch-event trigger condition: 0 = forward – the servo module looks for the actual position to change from less than the watch position to greater than the watch position. 1 = reverse – the servo module looks for the actual position to change from greater than the watch position to less than the watch position.
Position	REAL	Immediate or Tag	The new value for the watch position.

Structured Text

MAW(Axis,MotionControl,TriggerCondition,Position);

The operands are the same as those for the relay ladder MAW instruction.

Enter your selection for the operands that require you to select from available options.

This Operand	Has these options which you...	
	enter as text	or enter as a number
TriggerCondition	forward reverse	0 1

MOTION_INSTRUCTION Structure

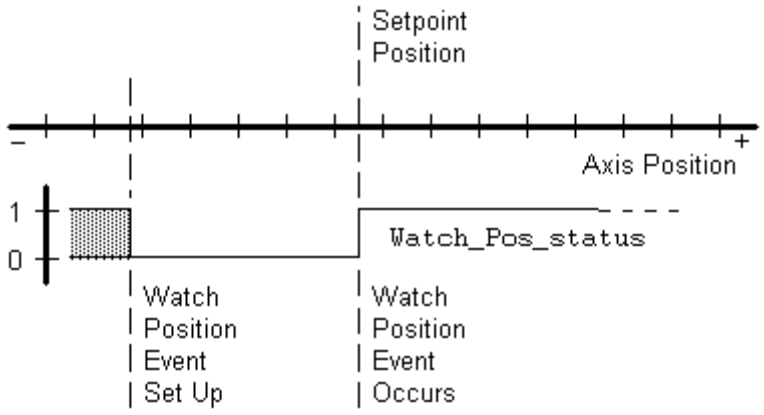
Table 86 - MAW MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis watch event checking has been successfully armed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the watch event has occurred, or has been superseded by another Motion Arm Watch, or terminated by a Motion Disarm Watch command.
.PC (Process Complete) Bit 27	It is set when a watch event occurs.

Description

The MAW instruction sets up a Watch Position event to occur when the specified physical axis reaches the specified Set-point position, as shown in Figure 18.

Figure 18 - Set Point Position



Watch Position events are useful for synchronizing an operation to a specified axis position while the axis is moving, such as activating a solenoid to push a carton off a conveyor at a certain axis position. Select or enter the desired physical axis, the desired Trigger Condition, and enter a value or tag variable for the desired Watch Position.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

When an Arm Watch Position instruction is executed, the WatchEventStatus bit is set to 0 (FALSE) and the actual position of a physical axis is monitored (at the servo loop update rate) until it reaches the specified Watch Position. After the watch position event occurs, the WatchEventStatus bit for the axis is set to 1 (TRUE).

Multiple watch position events can be active at a given time, however only one can be active at a time for any given physical axis. Each event is monitored independently and can be checked by using the appropriate WatchEventStatus bit.

IMPORTANT	In large I/O connections, force values can slow down the rate at which the controller processes repetitive watch positions.

Executing the Instruction

To successfully execute a MAW instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. The MAH instruction also applies to CIP axes configured for Feedback Only, Position Loop, Velocity Loop, or Torque Loop operation. Otherwise, the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

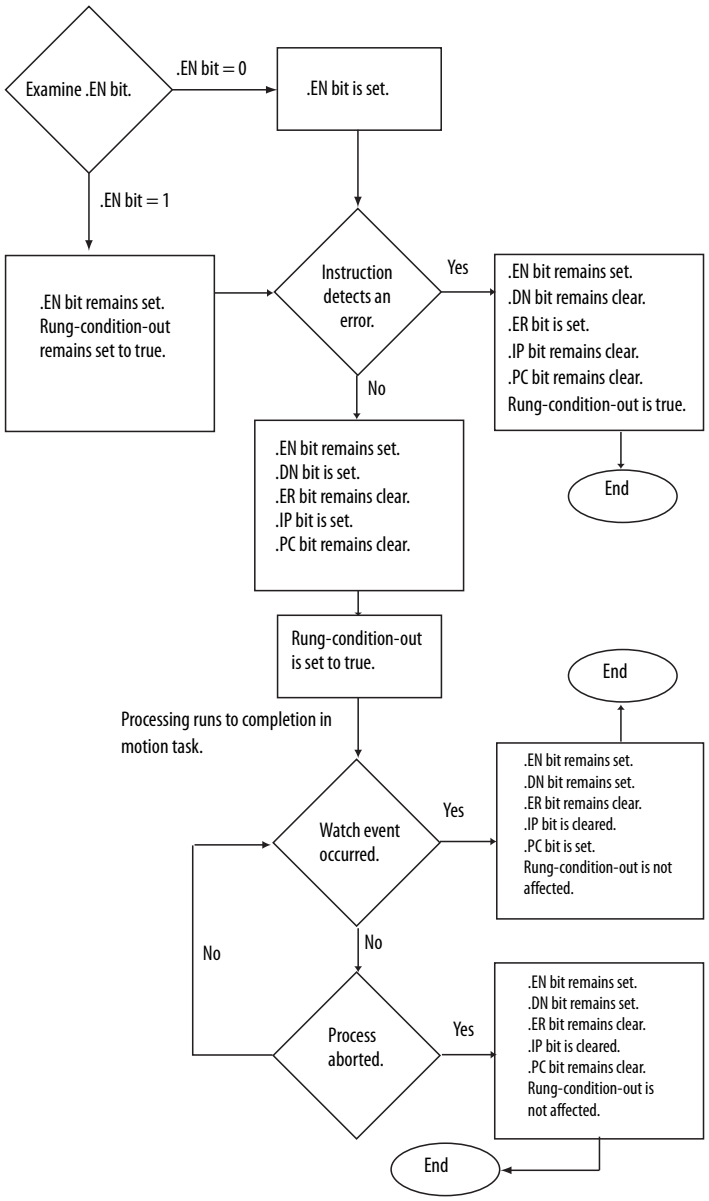
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAW Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.


Examine .EN bit.
.EN bit = 0
.EN bit is set.
.EN bit = 1
.EN bit remains set.
Rung-condition-out
remains set to true.
Instruction
detects an
error.
Yes
.EN bit remains set.
.DN bit remains clear.
.ER bit is set.
.IP bit remains clear.
.PC bit remains clear.
Rung-condition-out is true.
End
No
.EN bit remains set.
.DN bit is set.
.ER bit remains clear.
.IP bit is set.
.PC bit remains clear.
Rung-condition-out
is set to true.
Processing runs to completion in
motion task.
Watch event
occurred.
Yes
.EN bit remains set.
.DN bit remains set.
.ER bit remains clear.
.IP bit is cleared.
.PC bit is set.
Rung-condition-out is not
affected.
End
No
Process
aborted.
Yes
.EN bit remains set.
.DN bit remains set.
.ER bit remains clear.
.IP bit is cleared.
.PC bit remains clear.
Rung-condition-out is
not affected.
No
End

MAW Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAW Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MAW instruction receives a Servo Message Failure (12) error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No Resource (2)	Not enough memory resources to complete request. (sercos)

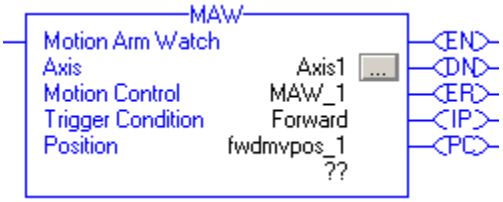
MAW Changes to Status Bits

Bit Name	State	Meaning
WatchEventArmedStatus	TRUE	The axis is looking for a watch position event.
WatchEventStatus	FALSE	The previous watch event is cleared.

MAW Example

When the input conditions are true, the controller arms watch-position event-checking for *axis1*.

Relay Ladder



Structured Text

```
MAW(Axis1,MAW_1,Forward,fwdmvpos_1);
```

Motion Disarm Watch (MDW)

Use the Motion Disarm Watch (MDW) instruction to disarm watch-position event-checking for an axis. This instruction has the affect of clearing both the Watch Event Status and Watch Armed Status bits in the axis data structure. Executing this instruction also clears the In Process bit associated with the controlling Motion Arm Watch (MAW) instruction.

Operands

The MDW instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 87 - MDW Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_ INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MDW(Axis,MotionControl);

The operands are the same as those for the relay ladder MDW instruction.

MOTION_INSTRUCTION Structure

Table 88 - MDW MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis watch event checking has been successfully disarmed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MDW instruction cancels watch position event checking set up by a previous Motion Arm Watch (MAW). The Disarm Watch Position instruction requires no parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

Executing the Instruction

To successfully execute a MDW instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. The MAH instruction also applies to CIP axes configured for Feedback Only, Position Loop, Velocity Loop, or Torque Loop operation. Otherwise, the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

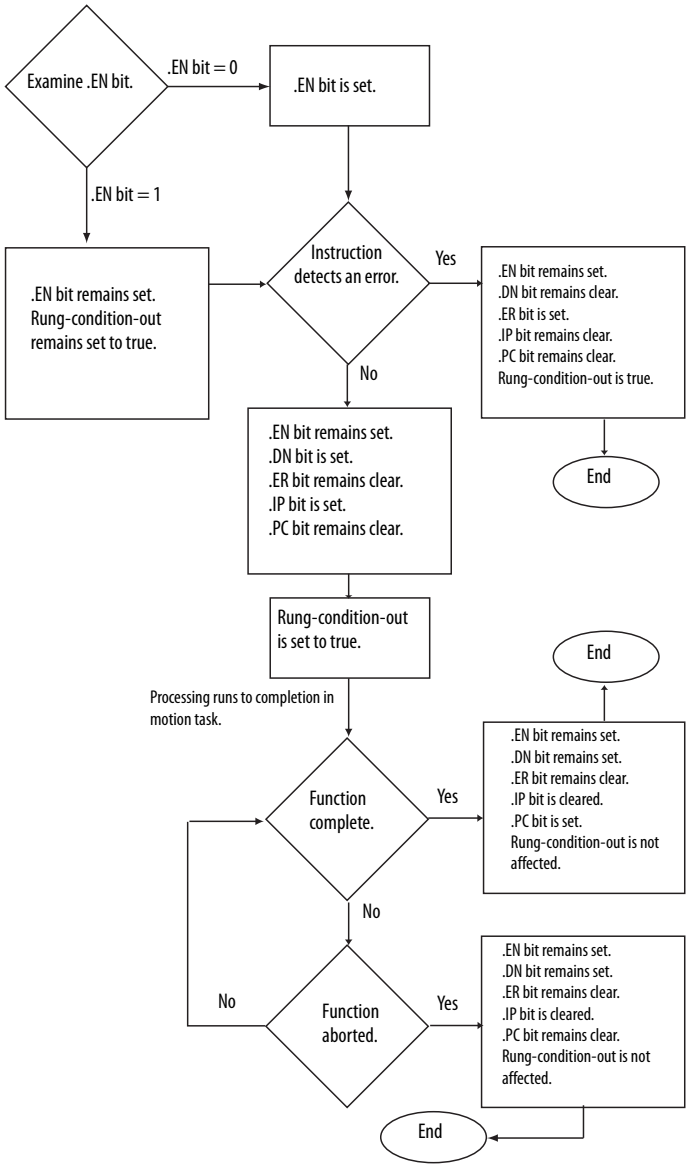
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MDW Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MDW Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

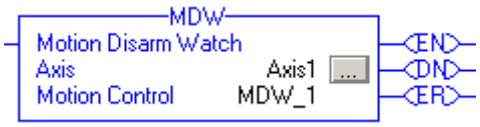
MDW Changes to Status Bits

Bit Name	State	Meaning
WatchEventArmedStatus	FALSE	The axis is not looking for a watch position event.
WatchEventStatus	FALSE	The previous watch event is cleared.

MDW Example

When the input conditions are true, the controller disarms watch-position event-checking for axis1.

Relay Ladder



Structured Text

```
MDW(Aaxis1,MDW_1);
```

Motion Arm Registration (MAR)

Use the Motion Arm Registration (MAR) instruction to arm servo module registration event checking for the specified axis. When the instruction is called, a registration event is armed based on the selected Registration Input and the specified Trigger Condition. When the specified Registration Input transition satisfies the Trigger Condition, the motion module computes the axis position at the moment the event occurred based on hardware latched encoder count data and stores it in the associated Registration Position variable in the axis data structure.

Also, the instruction's Event (PC) bit is simultaneously set, as well as the Registration Event Status bit in the axis data structure. If Windowed Registration is selected, only registration events whose computed registration position falls within the Max and Min Position window are accepted. If the Registration Position falls outside this window the registration event checking is automatically rearmed.

Operands

The MAR instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

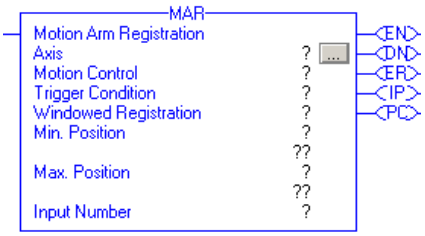


Table 89 - MAR Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Trigger condition	DINT	immediate	Defines the Registration Input transition that defines the registration event. 0 = trigger on positive edge. 1 = trigger on negative edge.

Table 89 - MAR Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Windowed registration	BOOLEAN	immediate	Enable (1) if registration is to be Windowed, that is, that the computed Registration Position must fall within the specified Min and Max Position limits to be accepted as a valid registration event. 0 = disabled. 1 = enabled.
Minimum position	REAL	immediate or Tag	Used when Windowed Registration is enabled. Registration Position must be greater than Min. Position limit before registration event is accepted.
Maximum position	REAL	immediate or Tag	Used when Windowed Registration is enabled. Registration Position must be less than Max. Position limit before registration event is accepted.
Input Number	DINT	1 or 2	Specifies the Registration Input to select. 1 = Registration 1 Position 2 = Registration 2 Position

Structured Text

MAR(Axis,MotionControl, TriggerCondition, WindowedRegistration, MinimumPosition, MaximumPosition, InputNumber);

The operands are the same as those for the relay ladder MAR instruction.

Enter your selection for the operands that require you to select from available options.

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
TriggerCondition	positive_edge negative_edge	0 1
WindowedRegistration	disabled enabled	0 1

MAR MOTION_INSTRUCTION Structure

Table 90 - MAR MOTION_INSTRUCTION Structure Descriptions

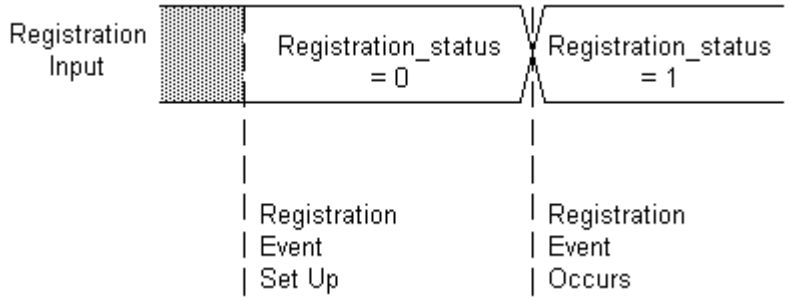
Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis registration event checking has been successfully armed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the registration event has occurred, or has been superseded by another Motion Arm Reg command, or terminated by a Motion Disarm Reg command.
.PC (Process Complete) Bit 27	It is set when a registration event occurs.

Description

The MAR instruction sets up a registration event to store the actual positions of the specified physical axis on the specified edge of the selected dedicated high speed Registration input for that axis.

When an MAR instruction is executed, the RegEventStatus bit is set to 0 (FALSE) and the selected Registration input for the specified axis is monitored by the motion module until a Registration input transition of the selected type (the registration event) occurs. When the registration event occurs, the RegEventStatus bit for the axis is set to 1 (TRUE) and the Actual Position of the axis is stored in the Registration Position variable corresponding to the registration input (for example, Registration 1 Position 1 or Registration 2 Position).

Figure 19 - Registration

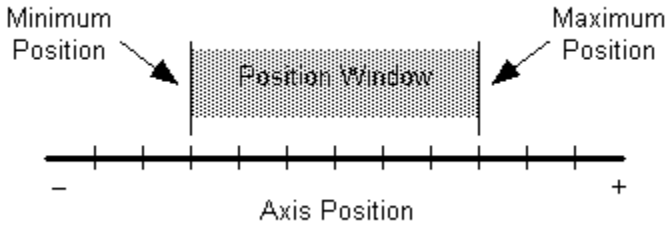


Multiple registration events can be active at any time for a given axis, but only one can be active per registration input. Each event is monitored independently and can be checked by using the appropriate RegEventStatus bit.

Windowed Registration

When the Windowed Reg checkbox is checked, the selected trip state only results in a registration event if it occurs when the axis is within the window defined by the minimum and maximum positions as shown in Figure 20.

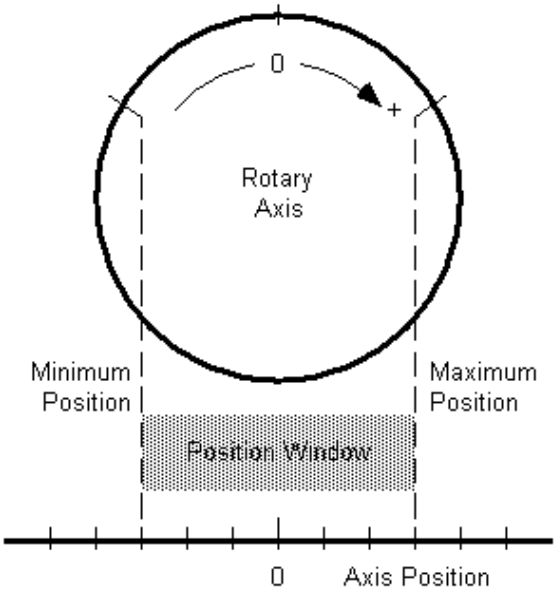
Figure 20 - Windowed Registration



Enter values or tag variables for the desired absolute positions that define the position window within which the selected trip state of the Registration input is valid. Windowed registration is useful in providing a mechanism to ignore spurious or random transitions of the registration sensor, thus improving the noise immunity of high-speed registration inputs.

For linear axes, the values can be positive, negative, or a combination. However, the Minimum Position value must be less than the Maximum Position value for the registration event to occur. For rotary axes, both values must be less than the unwind value set in the motion controller's machine setup menu. The Minimum Position value can be greater than the Maximum Position value for registration windows that cross the unwind point of the axis, as shown in Figure 21.

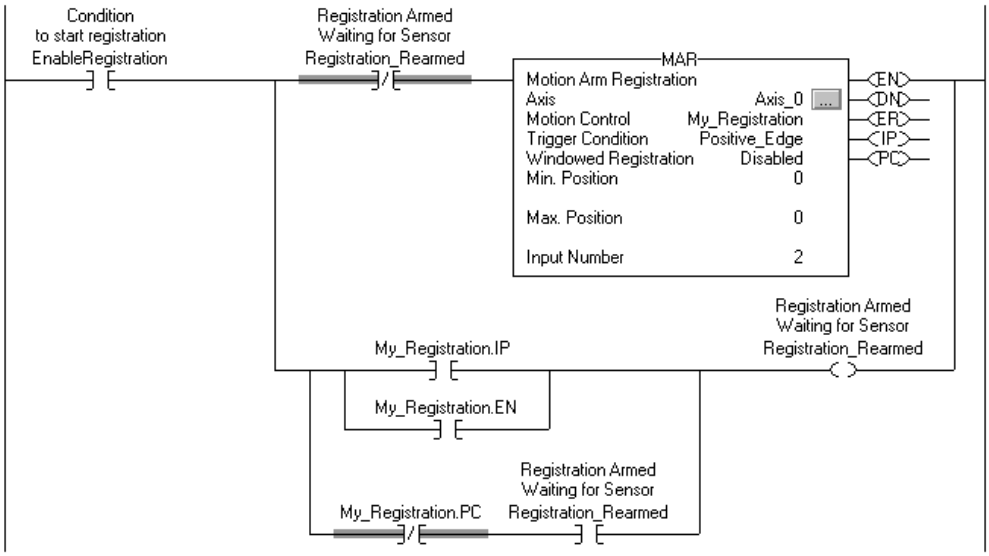
Figure 21 - Position Window for Rotary Axis



Rearming an MAR Instruction

If your application requires rapid and continuous detection of a registration sensor, we recommend that you use the following logic.

Figure 22 - Ladder Logic for Continuous Registration Detection



To rearm the MAR instruction, the rung must change from false to true. The rate at which this logic functions depends on the following:

- Program scan time
- Motion task course update rate

IMPORTANT	In large I/O connections, force values can slow down the rate at which the controller processes repetitive motion registration.

Executing the Instruction

To successfully execute a MAR instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. The MAH instruction also applies to CIP axes configured for Feedback Only, Position Loop, Velocity Loop, or Torque Loop operation. Otherwise, the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

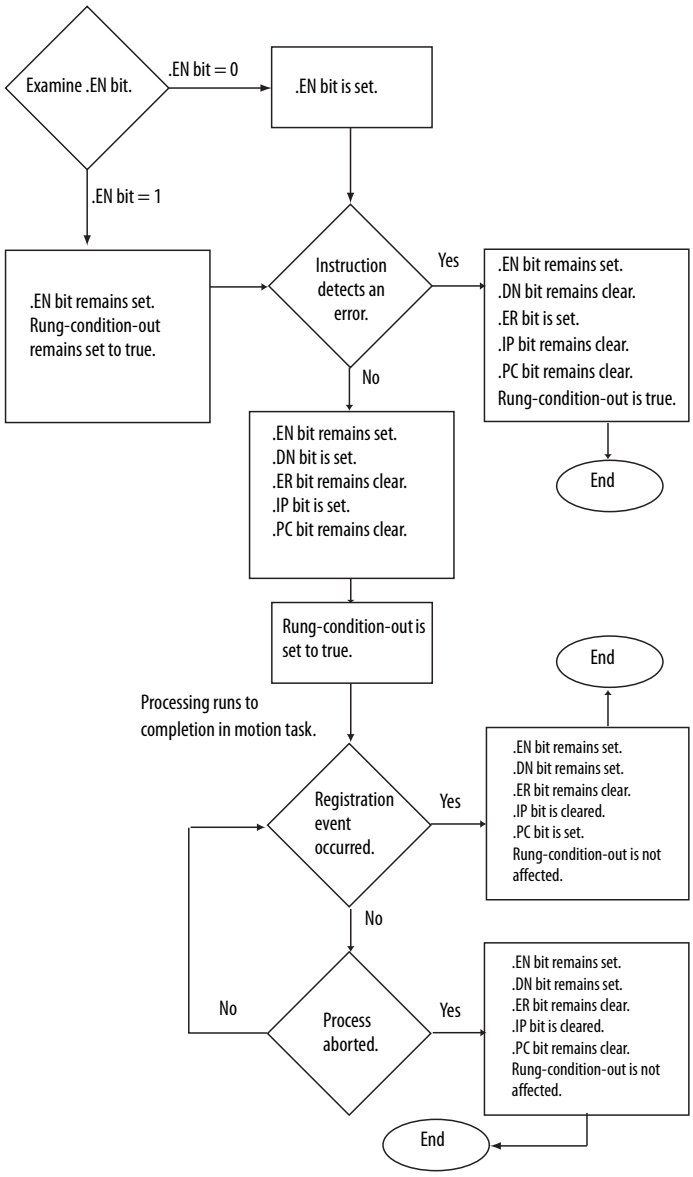
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAR Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See the following figure.	
postscan	The rung-condition-out is set to false.	No action taken.



MAR Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAR Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MAR instruction receives a Servo Message Failure (12) error message.

Table 91 - MAR Extended Error Codes Descriptions

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No Resource (2)	Not enough memory resources to complete request. (sercos)
SERVO_MESSAGE_FAILURE (12)	Invalid value (3)	Registration input provided is out of range.
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16).	Redefine Position, Home, and Registration 2 are mutually exclusive. (sercos)

Extended Error Codes for the Parameter Out of Range (13) error code work a little differently. Rather than having a standard enumeration, the number that appears for the Extended Error code refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MAR instruction, an extended error code of 4 would refer to the Min Position value. You would then have to check your value with the accepted range of values for the instruction.

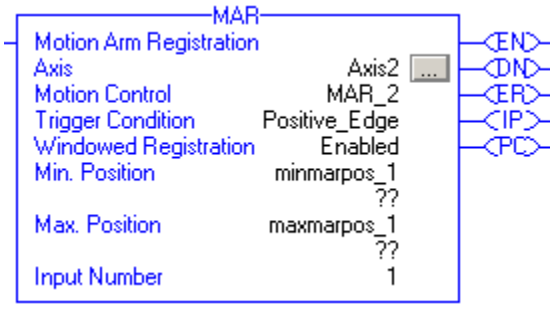
MAR Changes to Status Bits

Bit Name	State	Meaning
RegEventArmedStatus	True	The axis is looking for a registration event.
RegEventStatus	False	The previous registration event is cleared.

MAR Example

When the input conditions are true, the controller arms servo-module registration-event checking for *axis_0*.

Relay Ladder



Structured Text

```
MAR(Axis2,MAR_2,positive_edge,enabled,minmarpos_1,maxmarpos_1,1;
```

Motion Disarm Registration (MDR)

Use the Motion Disarm Registration (MDR) instruction to disarm the specified motion module registration input event checking for the specified axis. This instruction has the affect of clearing both the RegEventStatus and the RegArmedEventStatus bits. The In Process bit of the controlling Motion Arm Registration instruction, if any, is cleared as a result of executing the MDR instruction.

Operands

The MDR instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 92 - MDR Relay Ladder

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Input Number	DINT	1 or 2	Specifies the Registration Input to select. 1 = Registration 1 Position 2 = Registration 2 Position

Structured Text

MDR(Axis,MotionControl,InputNumber);

The operands are the same as those for the relay ladder MDR instruction.

MOTION_INSTRUCTION Structure

Table 93 - MAR Bit Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis watch event checking has been successfully disarmed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MDR instruction cancels registration event checking established by a previous Motion Arm Registration instruction. Only the registration checking associated with the specified registration input is disabled.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

Executing the Instruction

To successfully execute a MDR instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. The MAH instruction also applies to CIP axes configured for Feedback Only, Position Loop, Velocity Loop, or Torque Loop operation. Otherwise, the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MDR Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MDR instruction receives a Servo Message Failure (12) error message.

Table 94 - MDR Extended Error Code Descriptions

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Invalid value (3)	Registration input provided is out of range.

Extended Error Codes for the Parameter Out of Range (13) error code work a little differently. Rather than having a standard enumeration, the number that appears for the Extended Error code refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MDR instruction, an extended error code of 2 would refer to the Input Number operand's value. You would then have to check your value with the accepted range of values for the instruction.

MDR Changes to Status Bits

Bit Name	State	Meaning
RegEventArmedStatus	FALSE	The axis is not looking for a registration event.
RegEventStatus	FALSE	The previous registration event is cleared.

MDR Example

When the input conditions are true, the controller disarms registration-event checking for **axis_0**.

Relay Ladder

Structured Text

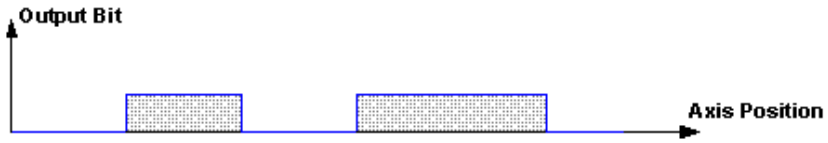
```
MDR(Axis2,MDR_1,2);
```

Motion Arm Output Cam (MAOC)

The Motion Arm Output Cam (MAOC) instruction offers the functionality to set and reset output bits based on an axis position. The MAOC instruction executes an output cam profile that is set up manually, programmatically, or by the Logix Designer Output Cam Editor. Internally, Output Cam objects handle Motion Planner Cam functionality. Each Output Cam object is responsible for one output, which consists of 32 output bits. Each single output bit can be programmed separately. Currently, Output Cam functionality is executed in the Logix controller every course update period (currently configurable between 1 and 32 ms).

For more information about how to program and understand scheduled outputs, see Position-based Output Control with the MAOC Instruction, publication 1756-AT017.

Figure 23 - Motion Planner Functionality



For an explanation of the differences between the Motion Planner in Logix Designer application versions 16 and 16.03, see Motion Planner, Version 16.03 Application Note, publication RA-AP031.

Operands

The MAOC instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

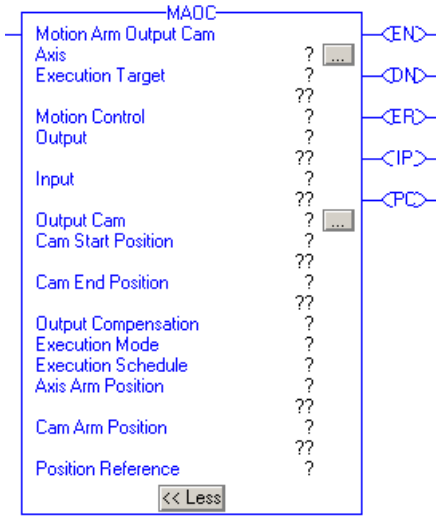


Table 95 - MAOC Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_CONSUMED AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis that provides the position input to the Output Cam. The axis can be a virtual, physical, or consumed axis.
Execution Target	DINT, SINT	Immediate or Tag	The execution target defines the specific Output Cam from the set connected to the named axis. Behavior is determined by the following: 0...8 = Output Cams executed in the Logix controller 9...31 = Reserved When you use the 1756-OB16IEFS scheduled output module, you can schedule all 16 outputs.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Output	DINT	Tag	A set of 32 output bits that are set or reset based on the specified Output Cam. It can be either a memory location or a physical output, for example, Local.0.O.Data. If Pending is selected as the Execution Schedule, then Output is ignored.
Input	DINT	Tag	A set of 32 input bits that can be used as enable bits depending on the specified Output Cam. It can be either a memory location or a physical input, for example, Local.0.I.Data. If Pending is selected as the Execution Schedule, then Input is ignored.
Output Cam	OUTPUT_CAM	Array Tag	An array of OUTPUT_CAM elements. The elements do not need to be ordered and the array size is determined by the number of cam elements specified. The array size is limited by the available memory of the Logix controller. For more information see Specifying Output Compensation on page 248.
Cam Start and Cam End Positions	SINT, INT, DINT, or REAL	Immediate or Tag	The cam start and cam end positions define the left and right boundary of the Output Cam range. When the cam position moves beyond the cam start or cam end position, the behavior of the Output Cam is defined by the execution mode and execution schedule. Changes to the cam start or cam end position don't take effect until the execution of a current MAOC instruction completes. If the latch or unlatch type is set to Position or Position and Enable with the enable bit active, the left and right cam positions specify the latch or unlatch position of the output bit.
Output Compensation	OUTPUT_COMPENSATION	Array Tag	Is an array of 1...32 OUTPUT_COMPENSATION elements. The array indices correspond to the output bit numbers. The minimum size of an array is determined by the highest compensated output bit. For more information see Specifying Output Compensation on page 248.

Table 95 - MAOC Relay Ladder Operand Descriptions (Continued)

Operand	Type	Format	Description
Execution Mode	DINT	Immediate	Depending on the selected execution mode, the Output Cam behavior can differ, when the cam position moves beyond the cam start or cam end position. The options are: 0 = Once Output Cam is disarmed and the Process Complete Bit of the Motion Instruction is set when the cam position moves beyond the cam start or the cam end position. 1 = Continuous Output Cam continues on the opposite side of the Output Cam range when the cam position moves beyond the cam start or the cam end position. 2 = Persistent Output Cam disarms when the cam position moves beyond the cam start or the cam end position. The Output Cam is rearmed when the cam position moves back into the Output Cam range.
Execution Schedule	DINT	Immediate	Selects when to arm the Output Cam. The options are: 0 = Immediate Output Cam is armed at once. 1 = Pending Output cam is armed when the cam position of a currently executing Output Cam moves beyond its cam start or cam end position. When Pending is selected the following parameters are ignored Output, Input, Axis Arm Position, and Reference. 2 = Forward only Output Cam is armed when the axis approaches or passes through the specified axis arm position in the forward direction. 3 = Reverse only Output Cam is armed when the axis approaches or passes through the specified axis arm position in the reverse direction. 4 = Bidirectional Output Cam is armed when the axis approaches or passes through the specified axis arm position in either direction.
Axis Arm Position	SINT, INT, DINT, or REAL	Immediate or Tag	This defines the axis position where the Output Cam is armed when the Execution Schedule is set to Forward Only, Reverse Only, or BiDirectional and the axis moves in the specified direction. If Pending is selected as the Execution Schedule, then Axis Arm Position is ignored. Changes to the axis arm or cam arm position only take effect after the execution of an MAOC instruction.
Cam Arm Position	SINT, INT, DINT, or REAL	Immediate or Tag	This defines the cam position associated with the axis arm position when the Output Cam is armed.
Reference	DINT	Immediate	Sets whether the Output Cam is connected to either Command position or Actual position of the axis. If Pending is selected as the Execution Schedule, then Reference is ignored. 0 = Actual The current position of the axis as measured by its encoder or other feedback device. 1 = Command The desired or commanded position of the master axis.

Structured Text

MAOC(Axis,ExecutionTarget,MotionControl,Output,Input,OutputCam,CamStartPosition,CamEndPosition,OutputCompensation,ExecutionMode,ExecutionSchedule,AxisArmPosition,CamArmPosition,Reference);

The operands are the same as those for the relay ladder MAOC instruction. For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]).

Table 96 - MAOC Operand Selections

This Operand	Has These Options That You...	
	Enter as Text	**Or Enter as a Number**
ExecutionMode	once continuous persistent	0 1 2
ExecutionSchedule	immediate pending forwardonly reverseonly bidirectional	0 1 2 3 4
Reference	actual command	0 1

MOTION_INSTRUCTION Structure

Table 97 - MAOC MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the rung goes false.
.DN (Done) Bit 29	It is set when Output Cam has been successfully initiated.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set when the Output Cam has been initiated successfully and cleared if either superseded by another Motion Arm Output Cam command, terminated by a Motion Disarm Output Cam command, or cam position moves beyond defined Output Cam range while execution mode is set to 'once'.
.PC (Process Complete) Bit 27	It is cleared on positive rung transition and set in once Execution Mode when cam position moves beyond defined Output Cam range.
.SEGMENT	It is set to the array index associated with error 36 (Illegal Output Cam) or error 37 (Illegal Output Compensation). Only the first of multiple errors is stored.

Description

Internally, Output Cam objects handle the Motion Planner Output Cam functionality. Each Output Cam object is responsible for one output, which consists of 32 output bits. Each single output bit can be programmed separately with an Output Cam profile, and compensated for position offset and time delay.

The MAOC instruction initiates the arming of a specific Output Cam between the designated axis and output. When executed, the specified output cam bits are synchronized to the designated axis by using an Output Cam Profile established by the Logix Designer Output Cam Editor. This relationship can be viewed as a master/slave relationship with the axis representing the master and the output bit representing the slave. Hence, the Output Cam functionality is related to the position cam functionality, which provides a relationship between a master axis and a slave axis.

To accurately synchronize the output cams to the designated axis, an execution schedule and associated axis and cam arm positions are specified. When the axis travels past the axis arm position in the direction specified by the Execution Schedule parameter, the cam position becomes locked to the axis position starting at the specified Cam Arm Position parameter. At this time the output cam is armed and the Output Cam Armed status is set. The output cam can also be configured via the Execution Schedule parameter to execute Immediately or Pending completion of a currently executing output cam. The output cam can also be executed Once, Continuously or Persistently by specifying the desired Execution Mode. Persistent behavior allows the output cam to become disarmed when the cam position exceeds the output cam range, and rearmed when cam position returns to within range.

MAOC Instruction

A valid Cam Arm position is any position, between and including, the Cam Start and Cam End positions. If the Cam Arm position is set to a value equal to (or very close to) the Cam Start or Cam End position, compensation can put a cam position out of range of the Cam Start and Cam End position. Compensation is affected by Output Compensation values specified for Position Offset, Latch Delay, and Unlatch Delay, as well as internal compensation values applied based on the Reference and Output parameters of the MAOC instruction.

No side affects occur if the MAOC instruction is configured with an Execution mode of Continuous or Persistent. A pending MAOC instruction does not exist when the Output Cam is armed and the axis moves.

The following side affects can occur of the MAOC instruction is configured with an Execution Mode of Once Only, and a pending MAOC exists when the Output Cam is armed and the axis moves.

- One or more outputs can never change state.
- The MAOC instruction can complete immediately.

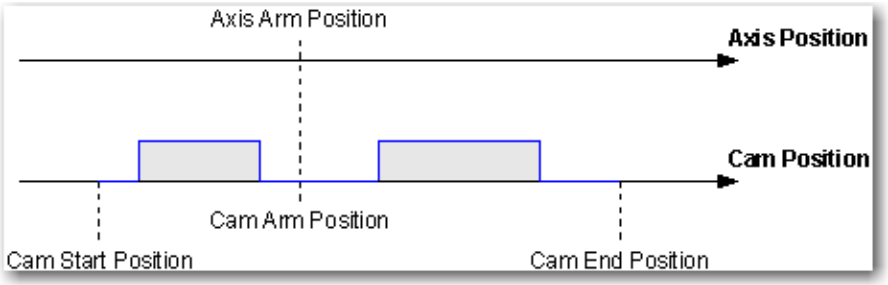
One possible side affect of a pending MAOC instruction existing when the Output Cam is armed and the axis moves is that one or more outputs could begin executing based on the configuration of the pending MAOC instruction.

Output Cam range is defined by input parameters CamStartPosition and CamEndPosition. The Master Reference selection allows axis input to be derived from either the Actual or Commanded position of the designated axis.

Axis Arm or Cam Arm Position

The axis arm position defines the axis position where the Output Cam is armed, if the execution schedule is set to either forward only, reverse only, or bidirectional and the axis moves in the specified direction. The cam arm position defines the cam position that is associated with the axis arm position, when the Output Cam is armed. Changes to the axis arm or cam arm position only take effect after the execution of an MAOC instruction.

Figure 24 - Axis Arm or Cam Arm Position



Reference

Depending on the selected reference, the Output Cam is connected to either the actual or command position of the axis.

IMPORTANT	Output cams increase the potential for exceeding coarse update rate. This can cause misbehavior if the motion task execution time exceeds the configured group coarse update period. The only way to check on this condition is to monitor the max execution time from the Motion Group Properties dialog box.

Executing the Instruction

IMPORTANT	The MAOC instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process .IP bit are set immediately. The In Process .IP bit remains set until the cam position moves beyond the cam start or cam end position in Once execution mode, is superseded by another MAOC instruction or is disarmed by the MDOC instruction. The Process Complete bit is cleared immediately when the MAOC executes and set when the cam position moves beyond the cam start or cam end position in Once execution mode.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Specifying the Output Cam

When you specify the Output Cam, you are determining latch and unlatch type behaviors, defining the range of an Output Cam element, and defining the cam duration.

To execute a MAOC instruction, a calculated Output Cam data array tag must be specified. Output Cam array tags can be created by the Logix Designer tag editor or the MAOC instruction by using the built-in Output Cam Editor. The data defines the specifics for each Output Cam element. The number of Output Cam elements is limited by the amount of available memory. Zero or more cams can be defined for each output bit. There is no constraint on how these elements are arranged within the Output Cam array.

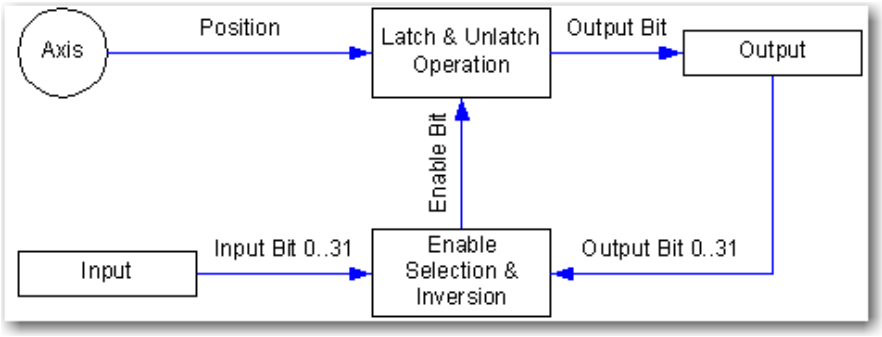
Refer to the description of the OUTPUT_CAM structure for more information about data types and programming units.

IMPORTANT	An anomaly occurs when the output CAM ON window positions are redefined while the output controlled by the output CAM element is active. In some instances, the Motion Planner can not detect the off-crossing of the window and the output controlled by the output CAM element remains ON. This issue applies to any output point or virtual output controlled by an MAOC instruction. Additionally, we recommend that you only change configuration when the CAM element is not active.

For more information about the OUTPUT_CAM data type structure, see OUTPUT_CAM Structure on page 357.

The following diagram shows the relationships between the axis, input, and output that are defined by the Output Cam element.

Figure 25 - Output Cam Relationships



Latch Type

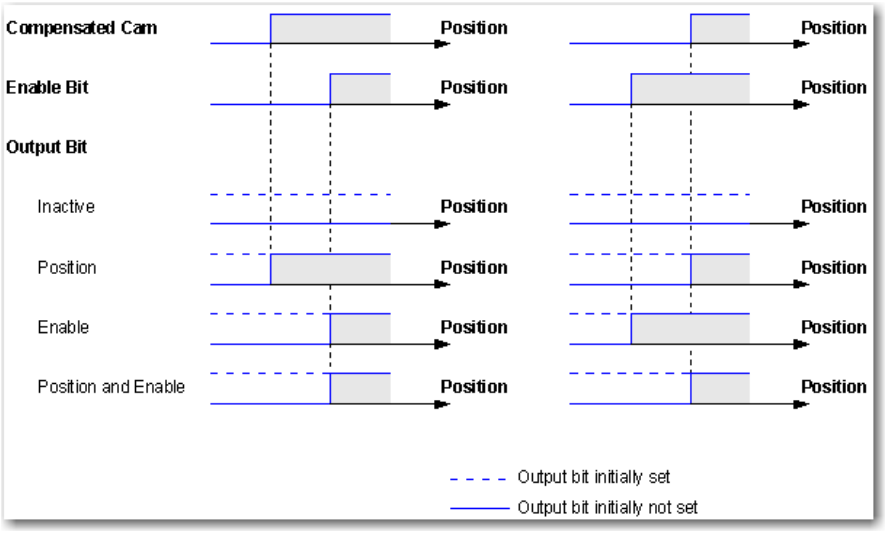
Depending on the selected LatchType, the corresponding output bit is set according to the following table.

Table 98 - Latch Type Behavior

Latch Type	Behavior
Inactive	The output bit is not changed.
Position	The output bit is set, when the axis enters the compensated cam range.
Enable	The output bit is set, when the enable bit becomes active.
Position and Enable	The output bit is set, when the axis enters the compensated cam range and the enable bit becomes active.

The following diagram shows the effect of the selected latch type on the output bit for different compensated cam and enable bit combinations as function of position.

Figure 26 - Compensated CAM and Enable Bit Combinations



Unlatch Type

Depending on the selected UnlatchType, the corresponding output bit is reset according to the following table.

Table 99 - Unlatch Type Behavior

Unlatch Type	Behavior
Inactive	The output bit is not changed.
Position	The output bit is reset when the axis enters the compensated cam range.
Duration	The output bit is reset when the duration expires.
Enable	The output bit is reset when the when the enable bit becomes inactive.
Position and Enable	The output bit is reset when the axis leaves the compensated cam range or the enable bit becomes inactive.
Duration and Enable	The output is reset when the duration expires or the enable bit becomes inactive.

Figure 27 shows the effect of the selected unlatch type on the output bit for different compensated cam and enable bit combinations as function of position.

Figure 27 - Effect of the Selected Unlatch Type on Position

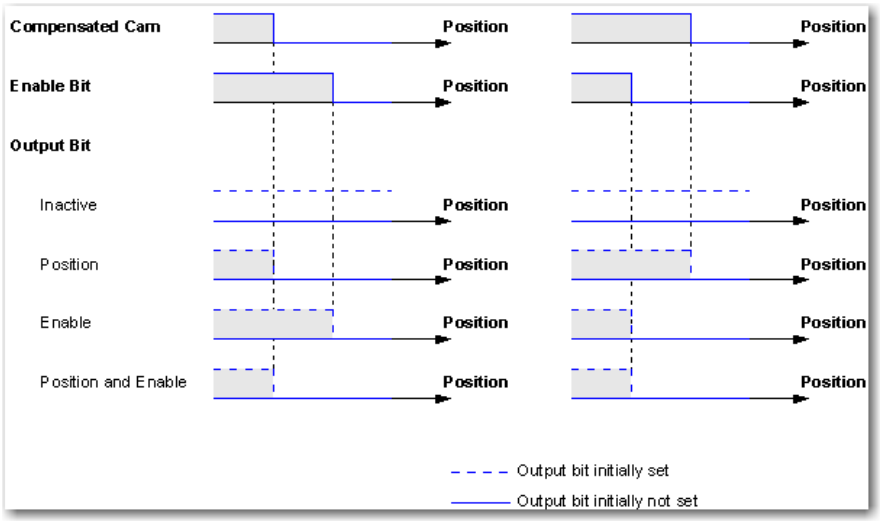
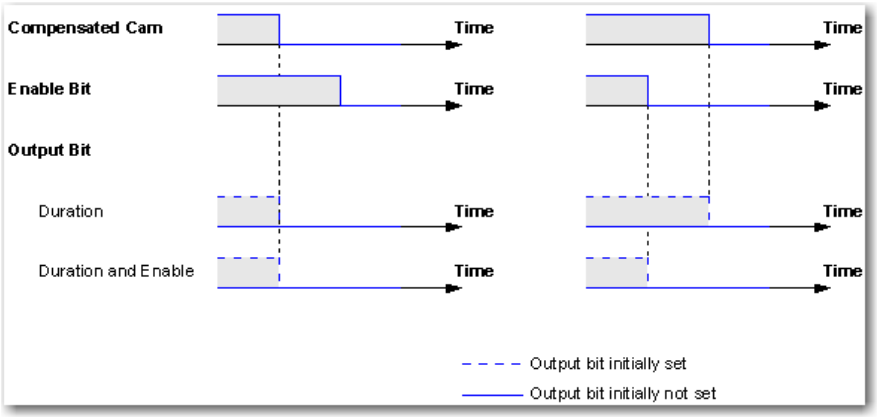


Figure 28 shows the effect of the selected unlatch type on the output bit for different compensated cam and enable bit combinations as function of time.

Figure 28 - Effect of the Selected Unlatch Type on Time



The following sections describe the affect of the Latch Type and Unlatch Type data types of the OUTPUT_CAM structure on the MAOC instruction. Refer to the description of the OUTPUT_CAM Structure on page 357 for more information on all of the data types and programming units.

Output Cam Array Checks

The following output cam array checks are used with the MAOC instruction.

Table 100 - Output Cam Array Checks

If you select	Then
An output bit less than 0 or greater than 31	The Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
A latch type less than 0 or greater than 3	A value of Inactive is used and the user is warned with an instruction error Illegal Output Cam.
An unlatch type less than 0 or greater than 5	A value of Inactive is used and the user is warned with an instruction error Illegal Output Cam.
A left or right position that is out of cam range and the latch or unlatch type is set to Position or Position and Enable	The Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
A duration that is less than or equal to 0 and the unlatch type is set to Duration or Duration and Enable	The cam start position is used and the user is warned with an instruction error Illegal Output Cam.
An enable type less than 0 or greater than 3 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable	The cam end position is used and the user is warned with an instruction error Illegal Output Cam.
An enable bit less than 0 or greater than 31 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable	The Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
A latch type that is set to Inactive and unlatch type is set to either Duration or Duration and Enable	The Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.

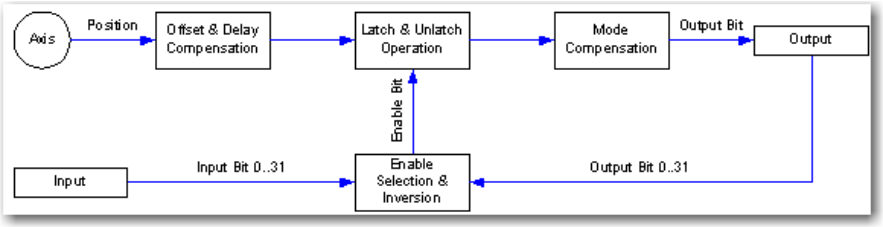
Specifying Output Compensation

An Output Compensation data array tag can be specified via the Logix Designer tag editor. The data type defines the specifics for each output bit by specifying the characteristics of each actuator. The array indices correspond to the output bit numbers. The number of the highest compensated output bit defines the minimum size of this array. Changes to the output compensation take effect immediately.

Refer to the description of the OUTPUT_CAM Structure on page 357 for more information on all of the data types and programming units.

Figure 29 shows the effect of the output compensation on the relationships between the axis, input, and output.

Figure 29 - Output Compensation Relationships

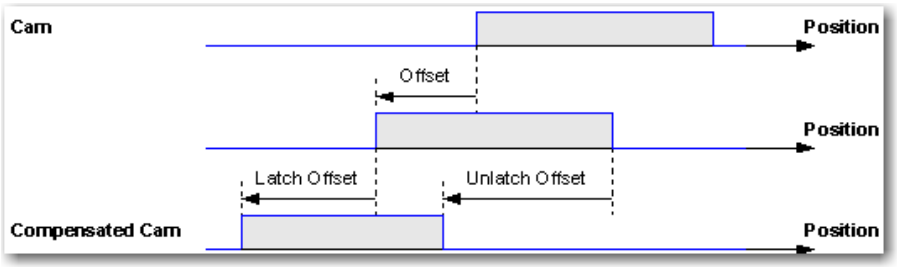


Specifying the output compensation includes setting the position, time, and mode compensation.

Offset and Delay Compensation

The offset provides position compensation, while the latch and unlatch delay provides time delay compensation for the latch and unlatch operation. Figure 30 shows the effect of the compensation values on an Output Cam element.

Figure 30 - Effects of Compensation Values



The cam range is defined by the left and right cam positions of the Output Cam element. The compensated cam range is defined by the cam range, offset, and latch and unlatch offsets. The latch and unlatch offsets are defined by the current speed v.

Latch Offset = v * Latch Delay

Unlatch Offset = v * Unlatch Delay

The resulting compensation offset can actually be larger than the difference between cam start and cam end position.

The following equation illustrates the effect of the compensation values on the duration of an Output Cam element.

Compensated Duration = Duration + Latch Delay - Unlatch Delay

Mode Compensation

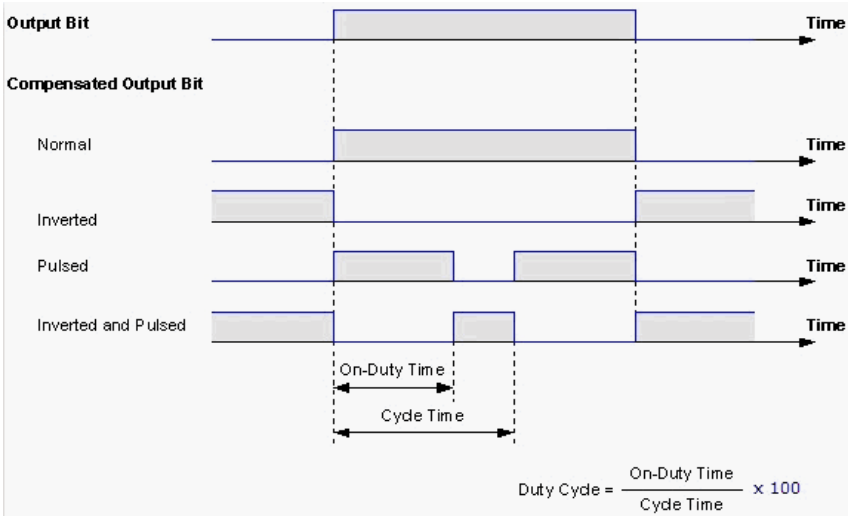
Depending on the selected mode, the compensated output bit is set according to this table.

Table 101 - Compensated Output Bit Behavior

Mode	Behavior
Normal	The output bit is set, when the output of the latch and unlatch operation becomes active. The output bit is reset, when the output of the latch and unlatch operation becomes inactive.
Inverted	The output bit is set, when the output of the latch and unlatch operation becomes inactive. The output bit is reset, when the output of the latch and unlatch operation becomes active.
Pulsed	The output bit is pulsed, when the output of the latch and unlatch operation is active. The on-duty state of the pulse corresponds to the active state of the output bit. The output bit is reset, when the output of the latch and unlatch operation becomes inactive.
Inverted and Pulsed	The output bit is pulsed, when the output of the latch and unlatch operation is active. The on-duty state of the pulse corresponds to the inactive state of the output bit. The output bit is set, when the output of the latch and unlatch operation becomes inactive.

Figure 31 shows the effect of the mode, cycle time, and duty cycle on an output bit.

Figure 31 - Mode Effects on an Output Bit



Output Compensation Array Checks

The following output compensation array checks are used with the MAOC instruction.

If you select	Then
A latch and unlatch delay combination that results in a compensated cam of less than minimum width	The width of the compensated cam is set to the minimum.
A mode less than 0 or greater than 3	A Normal mode is considered and you are warned with an instruction error Illegal Output Compensation.
A duty cycle less than 0 or greater than 100 and the mode is set to Pulsed or Inverted and Pulsed	A 0 or 100 duty cycle is considered and you are warned with an instruction error Illegal Output Compensation.
A cycle time less than or equal to 0 and the mode is set to Pulsed or Inverted and Pulsed	The output bit is not pulsed and you are warned with an instruction error Illegal Output Compensation.

Output

The output is the set of 32 output bits that can be set and reset depending on the specified Output Cam. The output can be either a memory location or a physical output, for example, Local.0.O.Data.

Input

The input is the set of 32 input bits that are can be used as enable bits depending on the specified Output Cam. The input can be either a memory location or a physical input, for example, Local.0.I.Data.

Arithmetic Status Flags

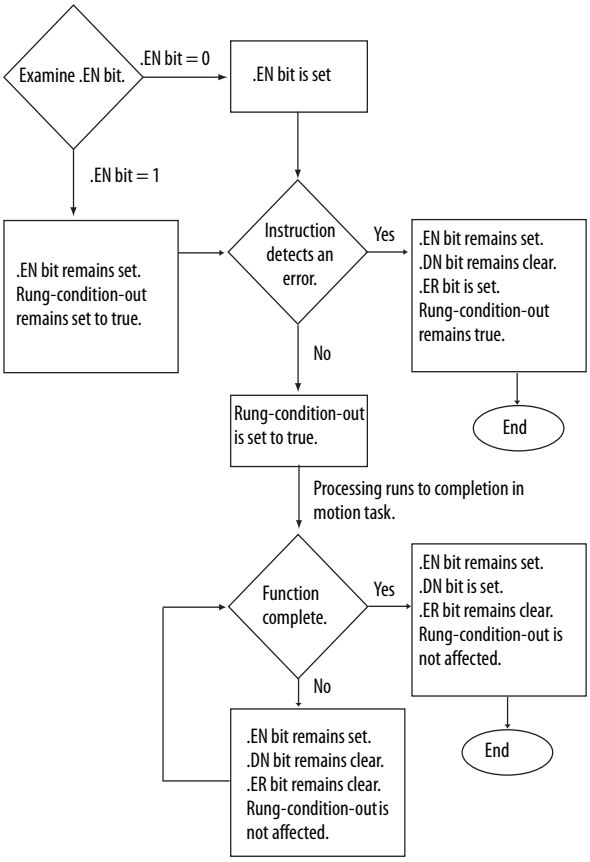
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAOC Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, and .ER bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution	See on page 252.	
postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAOC Extended Error Codes

Use Extended Error Codes (EXERR) for more information about an error. Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions.

Table 102 - MAOC Extended Error Codes Description

If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
07	Varies	Attempted execution.	Attempt execution with the axis in Shutdown state.	
11	Varies	Axis is not configured.	Reconfigure the axis.	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed
			4	Output
16	Varies	Homing is in a process error.	The axis is executing a homing sequence.	
19	Varies	Motion group is not synchronized.	Check the Motion group and make sure that the axis you are working with is associated to the group.	
20	Varies	Axis is in a faulted state.	Check the axis configuration.	
21	Varies	The motion group is in the faulted state	Check the Motion Group configuration.	
24	Varies	The controller is in an illegal controller mode.	Take the controller out of test mode.	
35	Varies	Illegal execution target	The specified execution target exceeds the number of Output Cam targets configured for the axis.	
36	Varies	The size of the Output Cam array is not supported or the value of at least one member is out of range.	Illegal Output Cam	Output bit less than 0 or greater than 31. Latch type less than 0 or greater than 3. Unlatch type less than 0 or greater than 5. Left or right position is out of cam range and the latch or unlatch type is set to Position or Position and Enable. Duration less than or equal to 0 and the unlatch type is set to Duration or Duration and Enable. Enable type less than 0 or greater than 3 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable. Enable bit less than 0 or greater than 31 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable. Latch type is set to Inactive and unlatch type is set to either Duration or Duration and Enable.
37	Varies	Either the size of the Output Compensation array is not supported or the value of one of its members is out of range.	Illegal Output Compensation	The array index associated with errors 36 and 37 are stored in .SEGMENT of the Motion Instruction data type. Only the first of multiple errors are stored. The specific error detected is stored in Extended Error Code. With the ability to dynamically modify the Output Cam table, the Illegal Output Cam error 36 can occur while the MAOC is in-process. In general, the cam elements in which an error was detected will be skipped. The following are exceptions, and will continue to be processed. Error 2, Latch Type Invalid. Latch Type defaults to Inactive. Error 3, Unlatch Type Invalid. Unlatch Type defaults to Inactive. Error 8, with Unlatch Type of Duration and Enable. Will behave as an Enable Unlatch type.
53	Varies	Inhibited axis.	Trying to initiate an MAOC on an inhibited axis.	

MAOC Changes to Status Bits

Status bits can be used to determine if an MAOC instruction can be initiated. The MAOC instruction affects the following status words in the Motion Axis Structure:

- OutputCamStatus
- OutputCamPendingStatus
- OutputCamLockStatus
- OutputCamTransitionStatus

If the Execution Schedule is set to Forward Only, Reverse Only or BiDirectional, an MAOC instruction can be initiated when either of the following two conditions exist:

- OutputCamStatus bit = FALSE
- OutputCamStatus bit = TRUE
  - OutputCamLockStatus bit = FALSE
  - OutputCamTransitionStatus bit = FALSE

Module Fault Conditions Disarm Output Cams

When the controller detects one of the following faults, it disarms output cams:

- For Axis_Servo and Axis_Servo_Drive, axis feedback loss fault
- For Axis_Servo and Axis_Servo_Drive, module fault
- For Axis_Consumed, physical axis fault

Those faults produce unreliable feedback data. Also, if an axis fault exists when an MAOC instruction is initiated, the instruction errs.

If the Execution Schedule is Pending, the MAOC instruction is initiated if either of the following two conditions exist:

- OutputCamStatus bit = FALSE
- OutputCamStatus bit = TRUE
  - OutputCamTransitionStatus bit = FALSE

Axis and Module Fault Conditions Disarm Output Cams

When the controller detects one of the following faults, it disarms output cams:

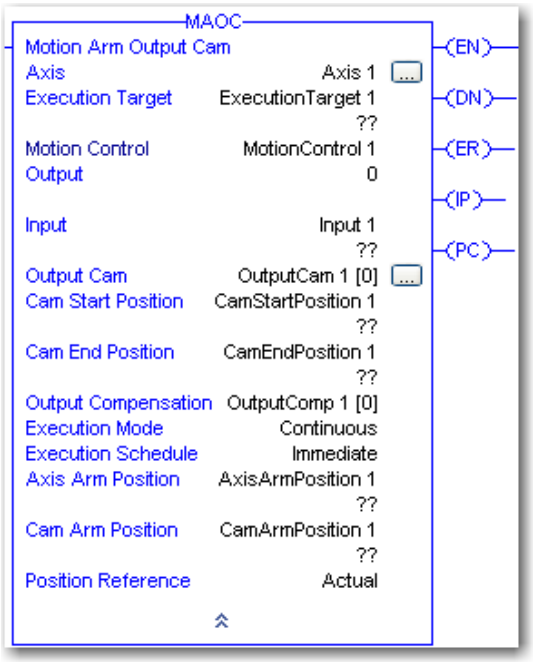
- for Axis_Servo and Axis_Servo_Drive, axis feedback loss fault.
- for Axis_Servo and Axis_Servo_Drive, module fault.
- for Axis_Consumed, physical axis fault.

Those faults produce unreliable feedback data.

Also, if an axis fault exists when an MAOC instruction is initiated, the instruction errs.

MAOC Example

Relay Ladder



Structured Text

MAOC(axis1,execution_target1,motion_control1,output1,input1,outputcam1[0],cam_start1,cam_end1,output_comp1[0],continuous,immediate,arm_position1,cam_arm_pos,actual);

The Scheduled Output module is designed to work in conjunction with the MAOC motion instruction.

Motion Disarm Output Cam (MDOC)

The Motion Disarm Output Cam (MDOC) instruction initiates the disarming of one or more Output Cams connected to the specified axis. Based on the disarm type, the MDOC instruction disarms either all Output Cams or only a specific Output Cam. The corresponding outputs maintain the last state after the disarming.

Operands

The MDOC instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

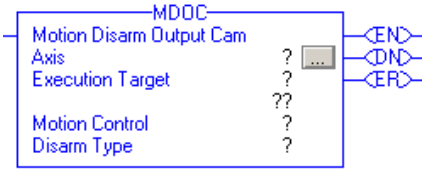


Table 103 - MDOC Relay Ladder Operand Description

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_FEEDBACK AXIS_CONSUMED AXIS_VIRTUAL AXIS_GENERIC AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis, which provides the position input to the Output Cam. Ellipsis launches Axis Properties dialog box.
Execution Target	SINT, INT, or DINT	immediate or Tag	The execution target defines the specific Output Cam from the set connected to the named axis. Behavior is determined by the following: • 0...8 – Output Cams executed in the Logix controller. • 9...31 – Reserved for future use.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Disarm Type	DINT	immediate	Selects one or all Output Cams to be disarmed for a specified axis. 0 = All – Disarms all Output Cams connected to the specified axis. 1 = Specific – Disarms the Output Cam that is connected to the specified axis and defined by the Execution Target.

Structured Text

MDOC(Axis,ExecutionTarget,MotionControl,DisarmType);

The operands are the same as those for the relay ladder MDOC instruction.

Enter your selection for the operands that require you to select from available options.

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
Disarm Type	all specific	0 1

MOTION_INSTRUCTION Structure

Table 104 - MDOC MOTION_INSTRUCTION Enumerations Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the rung goes false.
.DN (Done) Bit 29	It is set when the Output Cam(s) have been successfully disarmed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error.

Description

The MDOC instruction disarms a specific or all output cams for a specified axis depending on the selected disarm type. The axis provides the position input to the Output Cam. The execution target defines a specific Output Cam from the set that is connected to the specified axis.

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

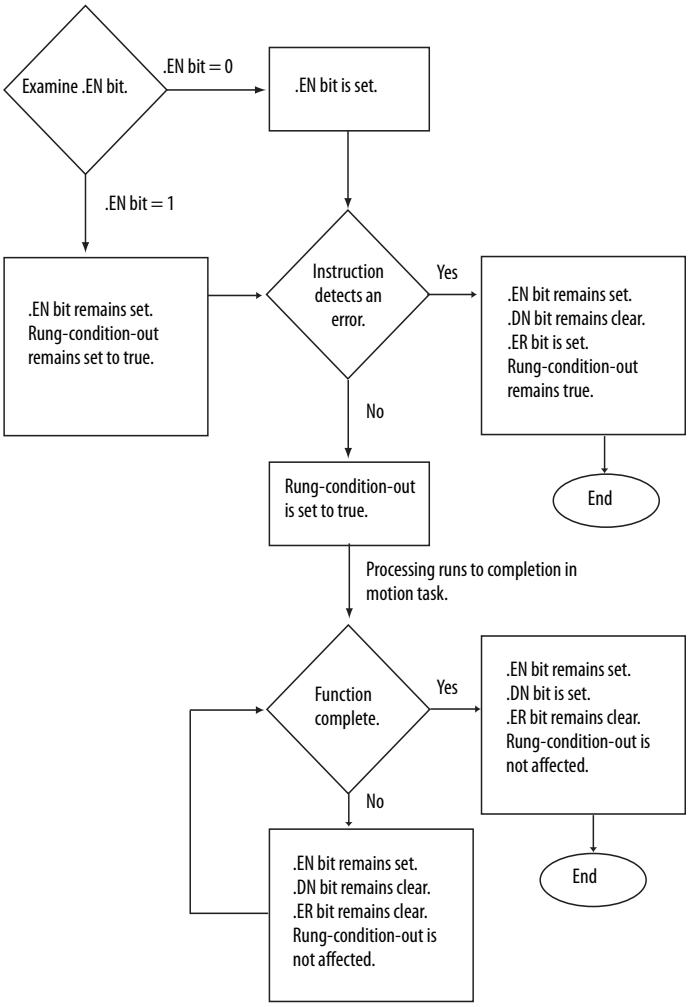
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MDOC Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
prescan	The .EN, .DN, and .ER bits are cleared. The rung-condition-out is set to false.	No action taken.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN and .ER bits are not affected. The rung-condition-out is set to false.	N/A
rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
instruction execution postscan	The rung-condition-out is set to false.	No action taken.



MDOC Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MDOC Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. Extended Error Codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MDOC instruction, an extended error code of 4 would refer to the Disarm Type operand's value. You would then have to check your value with the accepted range of values for the instruction.

Table 105 - MDOC Extended Error Codes Description

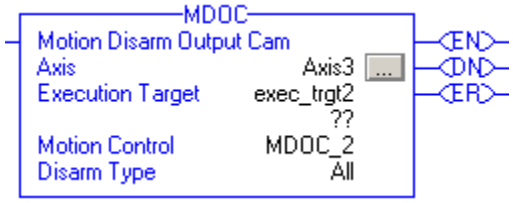
If ERR is	And EXERR is	Then		
		Cause	**Corrective Action**	
11	Varies	Axis is not configured.	Reconfigure the axis.	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	**Operand**
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed
			4	Output
19	Varies	Motion group is not synchronized.	Check the Motion group and make sure that the axis you are working with is associated to the group.	
35	Varies	Illegal execution target	The specified execution target exceeds the number of Output Cam targets configured for the axis.	
53	Varies	Inhibited axis.	Trying to initiate an MAOC on an inhibited axis.	

MDOC Changes to Status Bits

The MDOC instruction does not make any changes to the status bits.

MDOC Example

Relay Ladder



Structured Text

```
MDOC(Axis3,exec_trgt2,MDOC_2,all);
```

Motion Configuration Instructions
(MAAT, MRAT, MAHD, MRHD)

ATTENTION: Tags used for the motion control attribute of instructions should only be used once. Reuse of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

Configuration instructions include all motion instructions that are used to establish and apply servo configuration parameters to an axis. This group of instructions includes hookup test diagnostic instructions and tuning instructions.

Use the motion configuration instructions to tune an axis and to run diagnostic tests for the servo system. These tests include:

- a motor encoder hookup test.
- an encoder hookup test.
- a marker test
- a commutation test (for PM motors only)
- determine important motor parameters

These are the motion configuration instructions.

Table 106 - Choosing a Motion Configuration Instruction

If you want to	Use this instruction	Page	Languages
Compute a complete set of servo gains and dynamic limits based on a previously executed MRAT instruction. The MAAT instruction also updates the servo module with the new gain parameters.	Motion Apply Axis Tuning (MAAT)	261	Relay Ladder Structured Text
Command the servo module to run a tuning motion profile for an axis.	Motion Run Axis Tuning (MRAT)	268	
Apply the results of a previously executed MRHD instruction. The MAHD instruction generates a new set of encoder and servo polarities based on the observed direction of motion during the MRHD instruction.	Motion Apply Hookup Diagnostics (MAHD)	278	
Command the servo module to run one of three diagnostic tests on an axis.	Motion Run Hookup Diagnostics (MRHD)	284	

Motion Apply Axis Tuning (MAAT)

The Motion Apply Axis Tuning (MAAT) instruction is used to compute a complete set of servo gains and dynamic limits based on the results of a previously run Motion Run Axis Tuning (MRAT) instruction and update the motion

module with these new gain parameters. While this instruction takes no explicit parameters, input is derived from the Axis Tuning Configuration parameters as described in Tune Status Parameter on page 274. After execution of the MAAT instruction, the corresponding axis should be ready for servo activation.

Operands

The MAAT instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 107 - MAAT Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MAAT(Axis,MotionControl);

The operands are the same as those for the relay ladder MAAT instruction.

MOTION_INSTRUCTION Structure

Table 108 - MAAT MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN (Done) Bit 29	The done bit indicates when the instruction completes an apply axis-tuning process.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Description

The MAAT instruction is used to execute a series of computations resulting in values for gain and dynamic configuration parameters on the specified axis. As part of the work performed by MAAT, these resultant configuration parameters are applied so that the axis is ready for full servo operation. This instruction is designed to follow execution of the Motion Run Axis Tuning (MRAT) which generates axis input configuration values for the MAAT instruction. See the Motion Run Axis Tuning (MRAT) description for more information. MAAT requires no explicit input parameters; simply enter or select the desired physical axis. If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MAAT instruction uses axis configuration parameters as input and output. The input configuration parameters that MRAT uses are shown in the following table. Refer to the Motion Axis Object specification for a detailed description of these parameters.

The axis configuration parameters that MAAT uses as input depends on the External Drive configuration. If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, the following input parameters are required.

Axis Parameter	Data Type	Units	Descriptions
Tuning Velocity	Real	pos units/sec	Top Speed of Tuning Profile.
Tune Accel	Real	pos units/sec^2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec^2	Calculated Deceleration Time of Tuning Profile.
Tune Velocity Scaling	Real	mV/KCPS	Measured Velocity Scaling factor of axis Drive/Motor/ Encoder system.
Tune Velocity Bandwidth	Real	Hertz	Bandwidth of External Velocity Servo Drive

If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, the following input parameters are required.

Axis Parameter	Data Type	Units	Meaning
Damping Factor	Real	-	Damping Factor used to calculate the gains.
Tuning Velocity	Real	pos units/sec	Top Speed of Tuning Profile.
Tune Accel	Real	pos units/sec^2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec^2	Calculated Deceleration Time of Tuning Profile.
Effective Inertia	Real	mV/ KCPS^2	Computed Effective Inertia of Drive/ Motor system.
Position Servo Bandwidth	Real	Hertz	Maximum Position Servo Loop Bandwidth.

The axis configuration parameters that MAAT generates as output depend on the External Drive configuration. If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, the following output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Pos Proportional Gain	Real	1/msec	Position Servo Loop Proportional Gain
Pos Integral Gain	Real	$1/msec^2$	Position Servo Loop Integral Gain -- Set to Zero
Velocity Feedforward	Real	-	Position Servo Loop Proportional Gain
Acceleration Feedforward	Real	-	Velocity Command Feedforward -- Set to Zero
Max Speed	Real	pos units/sec	Maximum Speed for Motion Profiles – Set to Tuning Velocity
Max Acceleration	Real	pos units/sec^2	Maximum Acceleration for Motion Profiles
Max Deceleration	Real	pos units/sec^2	Maximum Acceleration for Motion Profiles
Output Filter Bandwidth	Real	Hertz	Bandwidth of Low Pass Servo Output Filter
Output Scaling	Real	mV/ KCPS	Scale Factor applied to output of the Position Servo Loop to the DAC.
Position Error Tolerance	Real	pos units	Maximum Servo Loop Position Error allowed without Fault.

If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, the following output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Pos Proportional Gain	Real	1/msec	Position Servo Loop Proportional Gain
Pos Integral Gain	Real	$1/msec^2$	Position Servo Loop Integral Gain
Vel Proportional Gain	Real	1/msec	Velocity Servo Loop Proportional Gain
Vel Integral Gain	Real	$1/msec^2$	Velocity Servo Loop Integral Gain
Velocity Feedforward	Real	-	Position Servo Loop Proportional Gain
Acceleration Feedforward	Real	-	Velocity Command Feedforward
Max Speed	Real	pos units/sec	Maximum Speed for Motion Profiles – Set to Tuning Velocity
Max Acceleration	Real	pos units/sec^2	Maximum Acceleration for Motion Profiles
Max Deceleration	Real	pos units/sec^2	Maximum Acceleration for Motion Profiles
Output Filter Bandwidth	Real	Hertz	Bandwidth of Low Pass Servo Output Filter
Output Scaling	Real	mV/ $KCPS^2$	Scale Factor applied to output of the Velocity Servo Loop to the DAC.
Position Error Tolerance	Real	pos units	Maximum Servo Loop Position Error allowed without Fault.

The previously listed output parameters generated by the MAAT instruction are immediately applied to the specified axis so that subsequent motion can be performed.

For more information about tuning configuration parameters see Tune Status Parameter on page 274.

Executing the Instruction

To successfully execute a MAAT instruction, the targeted axis must be configured as a Servo axis and be in the Axis Ready state, with servo action off. If these conditions are not met, the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

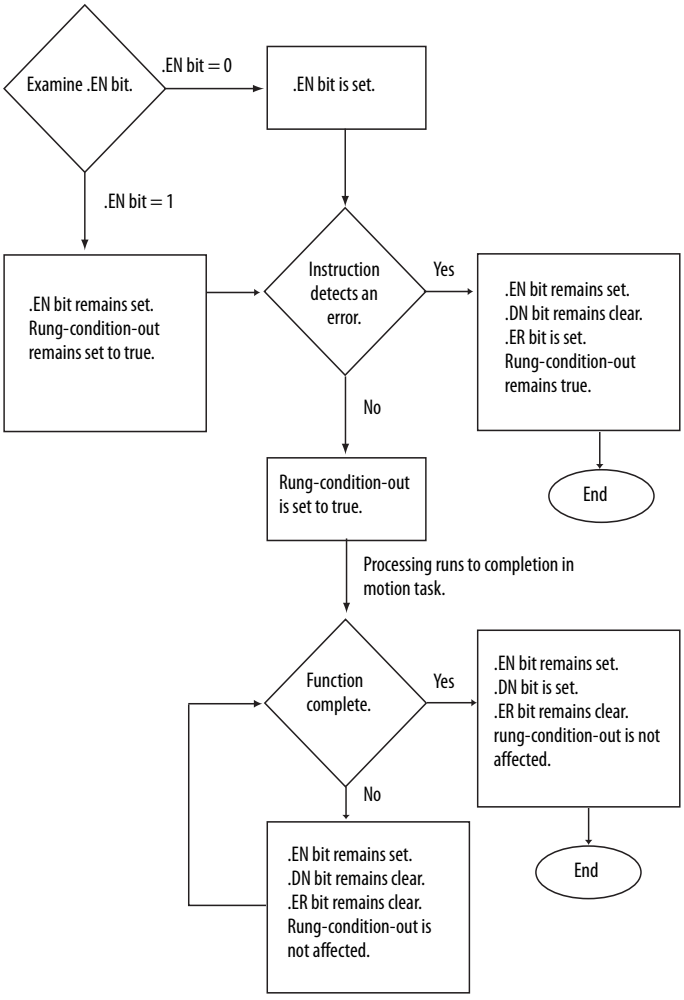
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAAT Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAAT Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MAAT instruction receives a Servo Message Failure (12) error message.

Table 109 - MAAT Extended Error Codes Meaning

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No Resource (2)	Not enough memory resources to complete request. (sercos)
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Permission denied (15)	Enable input switch error. (sercos)
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive (sercos), device state not correct for action. (sercos)

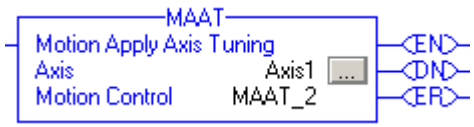
MAAT Changes to Status Bits

The MAAT instruction does not make any changes to the status bits.

MAAT Example

When the input conditions are true, the controller computes a complete set of servo gains and dynamic limits for axis1 based on the results of the previously executed Motion Run Axis Tuning (MRAT) instruction.

Relay Ladder



Structured Text

```
MAAT(Axis1,MAAT_2);
```

Motion Run Axis Tuning (MRAT)

Use the Motion Run Axis Tuning (MRAT) to command the motion module to run a tuning motion profile for the specified axis. The tuning motion profile consists of one or more acceleration and deceleration ramps induced by applying fixed voltages to the servo's drive output. Note that this instruction does not at any time close the servo loop. While this instruction takes no explicit input parameters, it does derive input from the Axis Tuning Configuration parameters. The result of executing the MRAT instruction is a set of measurement data that is stored in the Axis Object for subsequent use with the Motion Apply Axis Tuning (MAAT) instruction.

Operands

The MRAT instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

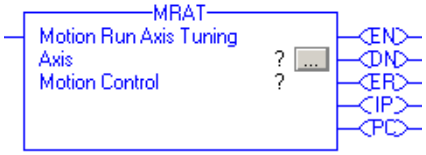


Table 110 - MRAT Relay Ladder Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

MRAT(Axis,MotionControl);

The operands are the same as those for the relay ladder MRAT instruction.

MOTION_INSTRUCTION Structure

Table 111 - MRAT MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set after the tuning process has been successfully completed.

Table 111 - MRAT MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the tuning process is complete, or terminated by a stop command, shutdown, or a servo fault.
.PC (Process Complete) Bit 27	It is set after the tuning process has been successfully completed.

Description - AXIS_SERVO, AXIS_SERVO_DRIVE

The MRAT instruction is used to execute a tuning motion profile on the specified axis. During this brief tuning motion profile, the motion module makes timing and velocity measurements that serve as input data for a subsequent Motion Apply Axis Tuning (MAAT) instruction. MRAT requires no explicit input parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRAT instruction uses axis configuration parameters as input and output. The input configuration parameters that MRAT uses are shown in this table.

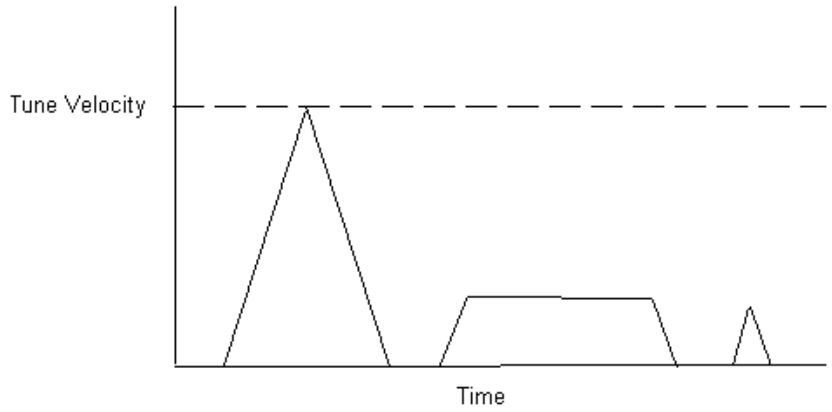
Table 112 - MRAT Input Configuration Parameters

Axis Parameter	Data Type	Units	Meaning
Tuning Direction	DINT	-	Direction of Tuning Motion (0-Fwd, 1-Rev)
Tuning Configuration Bits	DINT	-	Bit 0: Direction Bit 1: Position Integrator Bit 2: Velocity Integrator Bit 3: Velocity FF Bit 4: Acc FF Bit 5: Low Pass Filter Bit 6: Uni or Bi directional tune Bit 7: Friction Compensation
Tuning Travel Limit	Real	pos units	Maximum allowed excursion of Axis
Tuning Velocity	Real	pos units/sec	Top Speed of Tuning Profile.
Damping Factor	Real	-	Damping Factor used to calculate the maximum Position Servo Bandwidth.

Based on the configuration parameters in Table 112, MRAT execution generates a motion event on the specified axis that consists of a single triangular velocity profile or a series of three such profiles. Tune Velocity must be within the maximum speed capability of the drive and motor. The configured value for Tune Velocity should be set to the desired maximum operating speed of the axis so that the resulting tuning parameters are based on the dynamics of the system at that speed.

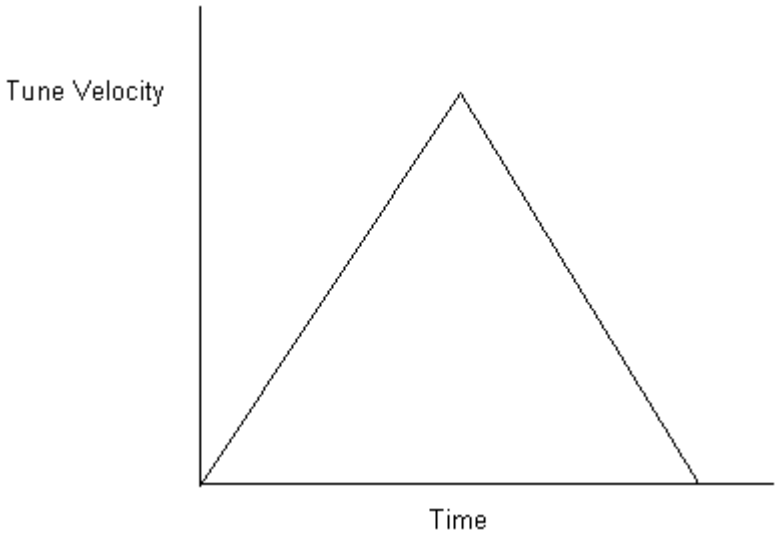
If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, three pulses are applied to the axis. The tuning velocity profile for this case is shown in Figure 32.

Figure 32 - Tuning Velocity Profile when True



If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, only one pulse is applied to the axis. The tuning velocity profile is shown in Figure 33.

Figure 33 - Tuning Velocity Profile when False



The axis configuration parameters that MRAT generates as output depend on the External Drive configuration. If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, the following output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Tune Status	Real	-	Status Report of the Tuning Process
Tune Accel Time	Real	seconds	Measured Acceleration Time of Tuning Profile.
Tune Decel Time	Real	seconds	Measured Deceleration Time of Tuning Profile.
Tune Accel	Real	pos units/sec^2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec^2	Calculated Deceleration Time of Tuning Profile.
Tune Velocity Scaling	Real	mV/KCPS	Measured Velocity Scaling factor of axis Drive/Motor/Encoder system.
Tune Rise Time	Real	mV/KCPS	Measured Rise Time of Tuning Step Response Profile.
Tune Velocity Bandwidth	Real	Hertz	Computed Bandwidth of External Velocity Servo Drive

If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, the following output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Tune Status	Real	-	Status Report of the Tuning Process
Tune Accel Time	Real	seconds	Measured Acceleration Time of Tuning Profile.
Tune Decel Time	Real	seconds	Measured Deceleration Time of Tuning Profile.
Tune Accel	Real	pos units/sec^2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec^2	Calculated Deceleration Time of Tuning Profile.
Effective Inertia	Real	mV/ KCPS^2	Computed Effective Inertia of Drive/ Motor system.
Position Servo Bandwidth	Real	Hertz	Calculated Maximum Position Servo Loop Bandwidth.

The previously listed output parameters generated by the MRAT instruction serve as inputs to a subsequent MAAT instruction which performs further tuning calculations and applies the results to various axis' servo and dynamic configuration parameters.

Description - AXIS_CIP_DRIVE

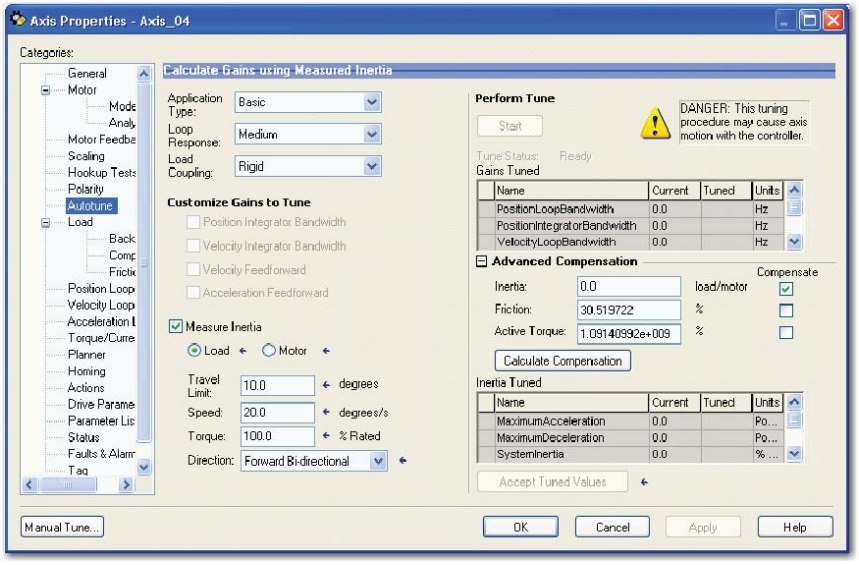
The MRAT instruction is used to execute a tuning motion profile on the specified CIP axis. MRAT requires no explicit input parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRAT instruction uses the CIP Axis configuration parameters as input and output. The input configuration parameters that MRAT uses are shown in this table.

Axis Parameter	Data Type	Units	Meaning
Tuning Direction	Short Integer	--	It determines the direction of the motion profile initiated by the Inertia Test service associated with the MRAT instruction. 0 = Unidirectional Forward 1 = Unidirectional Reverse 2 = Bidirectional Forward 3 = Bidirectional Reverse
Tuning Travel Limit	Real	Position Units	It is used by the Inertia Test service, associated with the MRAT instruction, to limit the excursion of the axis during the test
Tuning Speed	Real	Position Units/sec	The Tuning Speed attribute value determines the maximum speed used by the Inertia Test service initiated motion profile. This attribute should be set to the desired maximum operating speed of the motor prior to running the test.
Tuning Torque	Real	% Rated	It determines the maximum torque used by the Inertia Test service initiated motion profile. This attribute should be set to the desired maximum safe torque level prior to running the test. The default value is 100%, which yields the most accurate measure of the acceleration and deceleration capabilities of the system.
Damping Factor	Real	--	It is used in calculating the maximum Position and Velocity Servo Bandwidth values during execution of the MRAT instruction

The input configuration parameters can also be set in Axis Properties.



The Loop Response selection is used by the software to determine the value for the Damping Factor.

Loop Response	Damping Factor
Low	1.5
Medium	1.0
High	0.8

Based on the previously listed configuration parameters, MRAT execution generates a motion event on the specified axis that consists of a triangular velocity profile. The tuning procedure will measure maximum acceleration and deceleration rates based on ramps to and from the Tuning Speed. Thus, the accuracy of the measured acceleration and deceleration capability is reduced by tuning at a speed other than the desired operating speed of the system.

The axis configuration parameters that the MRAT instruction generates as output for CIP axes are shown in this table.

Axis Parameter	Data Type	Units	Meaning
Tune Status	Integer	--	The Tune Status attribute returns status of the last run Inertia Test service that initiates a process on the targeted drive axis
Tune Accel Time	Real	Seconds	Measured Acceleration time in seconds of the Tuning profile
Tune Decel Time	Real	Seconds	Measured deceleration time in seconds of the Tuning Profile
Tune Accel	Real	Position Units/sec^2	Measured Acceleration of the Tuning profile
Tune Decel	Real	Position Units/sec^2	Measured Deceleration of the Tuning profile
Tune Inertia Mass	Real	% Motor Rated / (Motor Units/Sec^2)	The estimated inertia or mass for the axis as calculated from the measurements made during the tuning process.
Tune Friction	Real	% Rated	The amount of friction measured during Tuning profile. This value can be used to configure the Friction Compensation feature of the drive.
Tune Load Offset	Real	% Rated	This value represents the active load offset measured during the Tune profile. This value can be used to set the Torque Offset of the drive to cancel out the active load torque/force.
Position Servo Bandwidth	Real	Hertz	It represents the unity gain bandwidth of the position loop that is used to calculate the position loop gains
Velocity Servo Bandwidth	Real	Hertz	It represents the unity gain bandwidth of the velocity loop that is used to calculate the velocity loop gains.

The previously listed output parameters generated by the MRAT instruction serve as inputs to compute the Position and Velocity loop gains, Position and Velocity Error Tolerances, Feed Forward Gains, Load Ratio, Maximum Acceleration, Maximum Deceleration, System Inertia, System Acceleration and Friction Compensation.

If the Gain Tuning Config Bits parameter bit zero is the Run Inertia Test Bit. This bit determines whether or not the MRAT tuning instruction will send a Test Inertia service to the drive to perform an inertia measurement. If this bit is set the Inertia Test shall be performed. If the bit is clear, the MRAT will immediately complete without an inertia measurement. It will only calculate the Pos and Velocity Servo Loop Bandwidths based on the Loop response or the Damping factor.

Tune Status Parameter

Conditions can occur that make it impossible for the controller to properly perform the tuning operation. When this is the case, the tuning process is automatically aborted and a tuning fault reported that is stored in the Tune Status output parameter (GSVable). It is also possible to manually abort a tuning process by using a Motion Axis Stop (MAS) instruction which results in a tuning fault reported by the Tune Status parameter. Possible values for Tuning Status are shown in this table.

Table 113 - Tuning Status Values

Status Code	Code	Meaning
Tune Success	0	Tune process has been successful.
Tune In Process	1	Tuning is in progress.
Tune Aborted	2	Tuning Process was aborted by user.
Tune Time-out	3	Tuning Process has timed out
Tune Servo Fault	4	Tuning Process Failed due to Servo Fault
Tune Travel Fault	5	Axis reached Tuning Travel Limit
Tune Polarity Fault	6	Axis motion heading in wrong direction due to incorrect motor/ encoder polarity configuration.
Tune Speed Fault	7	Axis tuning speed too low to achieve minimum measurement accuracy.
Tune Configuration Fault	8	The specified axis tuning configuration is not allowed and a fault occurs.

IMPORTANT The Tune Status Parameter is not to be mistaken for the .STATUS sub-tag of the MRAT instruction.

Executing the Instruction

To successfully execute a MRAT instruction on an axis, the targeted axis must be configured as a Servo Axis Type and the axis must be in the Axis Ready state. If any of these conditions are not met than the instruction errs.

IMPORTANT	When the MRAT instruction is initially executed the In Process (.IP) bit is set and the Process Complete (.PC) bit is cleared. The MRAT instruction execution can take multiple scans to execute because it requires transmission of multiple messages to the motion module. The Done (.DN) bit, is not set immediately, but only after these messages are successfully transmitted. The In Process (.IP) bit is cleared and the Process Complete (.PC) bit is set at the same time that the Done (.DN) bit is set.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

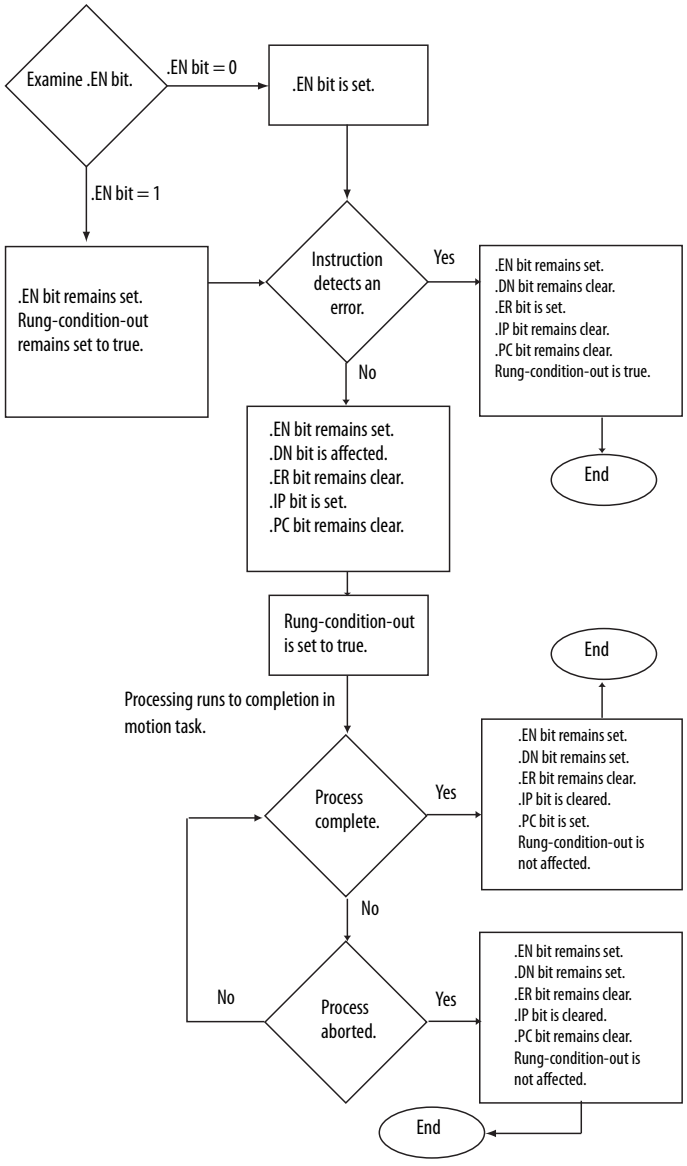
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MRAT Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MRAT Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MRAT instruction receives a Servo Message Failure (12) error message.

Table 114 - MRAT Extended Error Codes Meanings

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Process terminated on request (1)	Tune execution followed by an instruction to shutdown/disable drive, or a motion stop instruction or a Processor change requests a cancel of Tune.
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Incorrect Tune Process order. (sercos)

MRAT Changes to Status Bits

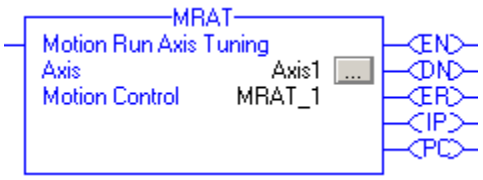
Bit Name	State	Meaning:
DriveEnableStatus	TRUE	Axis is in Drive Control state with the Drive Enable output active while the Tuning Profile is running.
TuneStatus	TRUE	The axis is running a tuning process.

MRAT Example

When the input conditions are true, the controller commands the servo module to run a tuning motion profile for *axis1*.

Relay Ladder

Figure 34 - MRAT Ladder Example



Structured Text

```
MAR(Axis1,MRAT_1);
```

Motion Apply Hookup Diagnostics (MAHD)

The Motion Apply Hookup Diagnostics (MAHD) instruction is used to apply the results of a previously run Motion Run Hookup Diagnostics (MRHD) instruction to generate a new set of encoder and servo polarities based on the Observed Direction of motion during the test. As part of the application process the instruction updates the motion module with these new polarity settings. After execution of the MAHD instruction, and assuming that a stable set of gains has been established, the corresponding axis should be ready for servo activation.

TIP The MAHD instruction is not supported for AXIS_CIP_DRIVE.

Operands

The MAHD instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

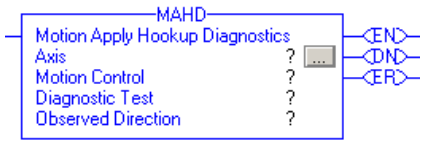


Table 115 - MAHD Relay Ladder Descriptions

Operand	Type	Format	Description
Axis	AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Diagnostic test	DINT	immediate	Selects the specific test for the motion module to run. 0 = motor/encoder hookup test 1 = encoder hookup test 2 = encoder marker test
Observed direction	DINT	immediate	Sets the direction of the test motion. 0 = forward. 1 = reverse.

Structured Text

MAHD(Axis,MotionControl,DiagnosticTest,ObservedDirection);

The operands are the same as those for the relay ladder MAHD instruction.

Enter your selection for the operands that require you to select from available options.

Table 116 - MAHD Structured Text Operands

This Operand	Has These Options Which You	
	Enter as Text	**Or Enter as a Number**
DiagnosticTest	motor_encoder encoder marker	0 1 2
ObservedDirection	forward reverse	0 1

MOTION_INSTRUCTION Structure

Table 117 - MAHD MOTION_INSTRUCTION Structure Descriptions

Enumerations	Description
.EN (Enable) Bit 31	Set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	Set after the hookup test apply process has been successfully executed.
.ER (Error) Bit 28	Set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MAHD instruction is used to execute a series of computations resulting in values for the Encoder Polarity and Servo Polarity configuration bit parameters of the specified axis. As part of work performed by MAHD, these resultant configuration bit parameters are applied to the motion module so that the axis is ready for full servo operation. This instruction is designed to follow execution of the Motion Run Hookup Diagnostics (MRHD) instruction which generates axis input configuration values for the MAHD instruction.

MAHD requires specification of the Diagnostic Test to apply and the Observed Direction of motion during the previous Motion Run Hookup Diagnostics (MRHD) instruction test process. Enter or select the Diagnostic Test and the Observed Direction and the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MAHD instruction uses axis configuration parameters as input and output. The input configuration parameters that MAHD uses are shown in the following table. The Test Direction Forward bit is automatically established as output from the Motion Run Hookup Diagnostics (MRHD) instruction.

Axis Parameter	Data Type	Units	Definition
Test Direction Forward	SINT	-	Direction of axis travel during hookup test as seen by the motion module.

Motor Encoder Hookup Test

If the Motor Encoder Test is selected, the controller computes the proper setting for both the Encoder Polarity and the Drive Polarity based on the Observed Direction instruction parameter and the state of Test Direction Forward bit which was established by the output of the Motion Run Hookup Diagnostics (MRHD) instruction. Once the Encoder Polarity and Drive Polarity settings are computed the MAHD applies these values to the corresponding axis configuration parameter bits as shown in the following table.

Axis Parameter	Data Type	Units	Definition
Encoder Polarity Negative	DINT	-	Inverts the sense of the encoder feedback input to the motion module.
Drive Polarity Negative	DINT	-	Inverts the sense of the DAC analog output from the motion module.

Encoder Hookup Test

If the Encoder Test is selected, the controller computes the proper setting for just the Encoder Polarity based on the Observed Direction instruction parameter and the state of Test Direction Forward bit which was established by the output of the Motion Run Hookup Diagnostics (MRHD) instruction. Once the Encoder Polarity and Drive Polarity settings are computed, the MAHD applies these values to the corresponding axis configuration parameter bits as shown in the following table.

Axis Parameter	Data Type	Units	Definition
Encoder Polarity Negative	DINT	-	Inverts the sense of the encoder feedback input to the motion module.

Executing the Instruction

To successfully execute a MAHD instruction running the Motor Encoder Test, the targeted axis must be configured as either a Servo or Feedback Only axis type. If any of these conditions are not met than the instruction errs.

IMPORTANT	The instruction execution can take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

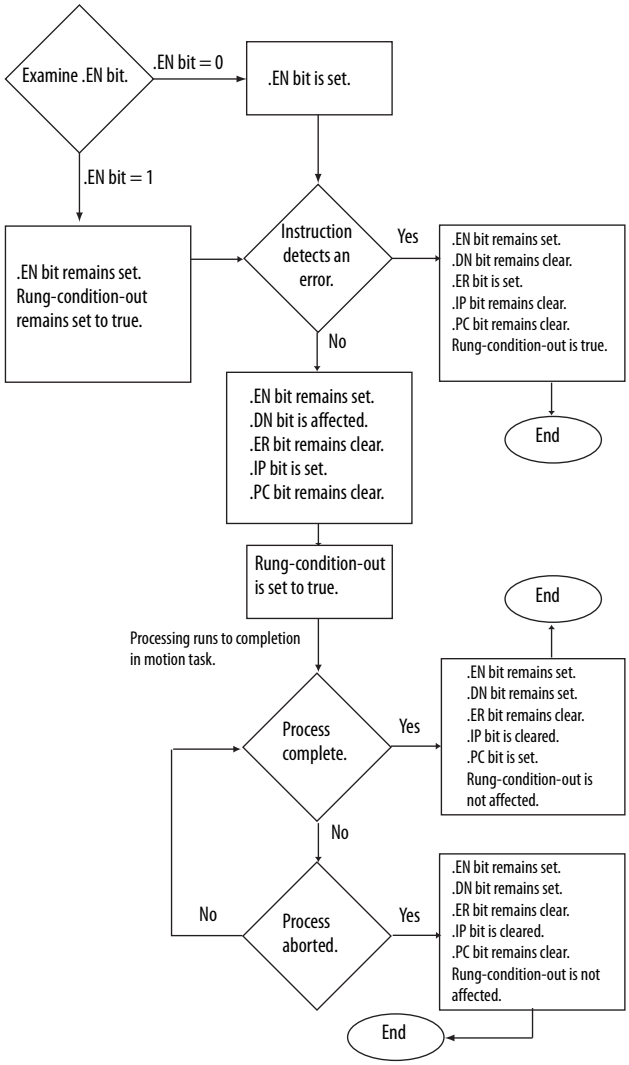
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MAHD Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



MAHD Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MAHD Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MAHD instruction receives a Servo Message Failure (12) error message.

Table 118 - MAHD Extended Error Codes Meanings

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No Resource (2)	Not enough memory resources to complete request. (sercos)
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Permission denied (15)	Enable input switch error. (sercos)
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive (sercos), device state not correct for action. (sercos)

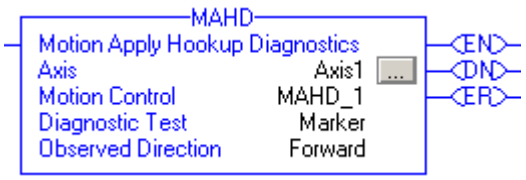
MAHD Changes to Status Bits

The MAHD instruction does not make any changes to the status bits.

Example

When the input conditions are true, the controller applies the results of a previously executed Motion Run Hookup Diagnostics (MRHD) instruction to axis1.

Relay Ladder



Structured Text

```
MAHD(axis1,axis1_MAHD,marker,forward);
```

Motion Run Hookup Diagnostics (MRHD)

Use the Motion Run Hookup Diagnostics (MRHD) instruction to command the motion module to run any one of three different diagnostics on the specified axis as selected by the Test ID. Currently diagnostics are available to test the motor/encoder hookup for a servo axis, the encoder hookup only, and the encoder marker hookup. Only the motor/encoder diagnostic initiates motion on the axis. This action consists of a short move of a user Motor Encoder Test Increment. The move is initiated by roughly 1 Volt per second ramping level of the servo's drive output. The result of executing the MRHD instruction is that the parameters, Test Status and Test Direction Forward are updated.

Operands

The MRHD instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

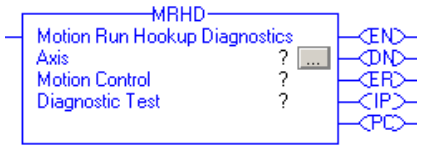


Table 119 - MRHD Operand Descriptions

Operand	Type	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on.
Motion control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Diagnostic test	DINT	immediate	Selects the specific test for the motion module to run. 0 = motor/encoder hookup test 1 = encoder hookup test 2 = encoder marker hookup test 3 = commutation test

Structured Text

MRHD(Axis,MotionControl,DiagnosticTest);

The operands are the same as those for the relay ladder MRHD instruction.

For the operands that require you to select from available options, enter your selection as:

Table 120 - MRHD Structured Text Operand

This Operand	Has these options which you	
	Enter as text	**Or enter as a number**
DiagnosticTest	motor_encoder	0
	encoder	1
	mark	2
	commutation	3

MOTION_INSTRUCTION Structure

Table 121 - MRHD MOTION_INSTRUCTION Structure Descriptions

Bit	Enumerations	Description
26	.IP	In Process is set on positive rung transition and cleared after the diagnostic test process is complete, or terminated by a stop command, shutdown, or a servo fault.
27	.PC	Process Complete is set after the diagnostic test process has been successfully completed.
28	.ER	Error is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
29	.DN	Done is set after the hookup test process has been successfully executed.
31	.EN	Enable is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.

Description - AXIS_SERVO, AXIS_SERVO_DRIVE

The MRHD instruction is used to execute various test diagnostics on the specified axis to test the integrity and, in some cases, the polarity of servo field connections. There are currently test diagnostics supporting drive hookup, encoder hookup, marker hookup and motion module OK contact hookup. During some of these test processes the motion module generates output to the external drive to produce a small amount of motion. Measurements made during some of these hookup diagnostic tests are saved as output configuration parameters that also serve as input data for a subsequent MAHD (Motion Apply Hookup Diagnostic) instruction. MRHD require only one explicit input parameter, Diagnostic Test. Enter or select the Diagnostic Test to run and the axis to test.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRHD instruction uses axis configuration parameters as input and output. The input configuration parameters that MRHD uses are shown in the following table.

Axis Parameter	Data Type	Units	Definition
Motor Encoder Test Increment	Real	-	Distance that the Axis must travel to satisfy the Hookup Diagnostic Test

The axis configuration parameters that MRHD generates as output depend on the specified Hookup Diagnostic.

Motor Encoder Hookup Test

If the Motor Encoder Test is selected, the motion module enables the external drive and generates a 1 Volt per second output ramp to the drive while monitoring the encoder feedback. When the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test voltage is set back to zero and the drive disabled. The motion module then reports the direction of travel which is stored as one of the following output parameters.

Axis Parameter	Data Type	Units	Definition
Test Status	Integer	-	Status Report of the Hookup Diagnostic Test Process
Test Direction Forward	DINT	-	Direction of axis travel during hookup test as seen by the motion module.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect that axis reaching the configured Motor Encoder Test Increment within 2 seconds, the servo sets the test voltage back to zero and disables the drive. The control reflects this condition through the Test Status axis output parameter. This usually indicates that either the cabling to the drive or the cabling to the encoder is incorrect. Running MRHD with the Encoder Hookup Test selected is an effective method of isolating the problem to the encoder or drive.

Encoder Hookup Test

If the Encoder Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects that the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test is complete. The motion module then reports the direction of travel as one of the following MRHD output parameters.

Axis Parameter	Data Type	Units	Definition
Test Status	Integer	-	Status Report of the Hookup Diagnostic Test Process
Test Direction Forward	DINT	-	Direction of axis travel during hookup test as seen by the motion module.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect the axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test by using the MAS instruction and check the encoder wiring.

Marker Hookup Test

If the Marker Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects a marker (Channel Z) pulse, the test is then complete. The motion module then reports success via the Test Status.

Axis Parameter	Data Type	Units	Definition
Test Status	Integer	-	Status Report of the Hookup Diagnostic Test Process
Test Direction Forward	DINT	-	Direction of axis travel during hookup test as seen by the motion module.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect that axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test by using the MAS instruction and check the encoder wiring.

Test Status

Conditions can occur that make it impossible for the control to properly perform the test operation. When this is the case, the test process is automatically aborted and a test fault is reported and stored in the Test Status output parameter. It is also possible to manually abort a test process by using a MAS instruction which results in a test fault reported by the Test Status parameter. Possible values for Test Status are shown in this table.

Table 122 - Test Status Values

Error Message	Code	Definition
Test Success	0	Test process has been successful.
Test In Process	1	Test is in progress.
Test Aborted	2	Test Process was aborted by user.
Test Time-out	3	Test Process has exceeded timed out (2 Seconds)
Test Servo Fault	4	Test Process Failed due to Servo Fault
Test Increment Fault	5	Test Process Failed due to insufficient test increment distance to make a reliable measurement.
Test Increment Fault Feedback 2	6	Test Process Failed due to insufficient test increment distance to make a reliable measurement on Feedback 2.

To successfully execute a MRHD instruction running the Motor Encoder Test, the targeted axis must be configured as a Servo Axis Type and the axis must be in the Axis Ready state. For other tests this instruction executes properly on either a Servo or Feedback Only axis type. If any of these conditions are not met than the instruction errs.

IMPORTANT	When the MRHD instruction is initially executed the In process (.IP) bit is set and the Process Complete (.PC) bit is cleared. The MRHD instruction execution can take multiple scans to execute because it requires transmission of multiple messages to the motion module. The Done (.DN) bit, is not set immediately, but after these messages are successfully transmitted. The In process (.IP) bit is cleared and the Process Complete (.PC) bit is set at the same time that the Done (.DN) bit is set.

Description - AXIS_CIP_DRIVE

The MRHD instruction is used to execute various test diagnostics on the specified CIP axis to test the integrity and, in some cases, the polarity of servo field connections. There are currently test diagnostics supporting drive hookup, encoder hookup, marker hookup and motor commutation. During some of these test processes the motion module generates output to the external drive to produce a small amount of motion. Measurements made during some of these hookup diagnostic tests are saved as output configuration parameters. MRHD requires only one explicit input parameter, Diagnostic Test. Enter or select the Diagnostic Test to run and the axis to test.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRHD instruction uses the CIP axis configuration parameters as input and output. The input configuration parameters that the MRHD uses are shown in the following table.

Axis Parameter	Data Type	Units	Definition
Hookup Test Distance	Real	Position Units	Distance that the axis must travel to satisfy the selected hookup test process
Hookup Test Time	Real	Seconds	Time that the axis must continue moving to satisfy the selected hookup test
Hookup Test Feedback Channel	USINT	-	Used by the Hookup Test service when the `encoder' test is selected to determine which feedback channel to test 1 = Feedback 1 2 = Feedback 2

The CIP axis configuration parameters that MRHD generates as output depend on the specified Hookup Diagnostic.

Motor Encoder Hookup Test

If the Motor Encoder Test is selected, the motion module enables the external drive and generates a 1 Volt per second output ramp to the drive while monitoring the encoder feedback. When the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test voltage is set back to zero and the drive disabled. The motion module then reports the direction of travel which is stored as one of the following output parameters.

Table 123 - MRHD Axis Parameter Descriptions

Axis Parameter	Data Type	Units	Descriptions
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions can occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7…255 = reserved
Hookup Test Feedback Direction 1	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 1 device. 0 = The drive's feedback 1 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 1 device detected a negative direction, that is, decreasing counts. 2…255 = reserved The value for Hookup Test Feedback 1 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

Table 123 - MRHD Axis Parameter Descriptions

Axis Parameter	Data Type	Units	Descriptions
Hookup Test Feedback Direction 2	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 2 device. 0 = The drive's feedback 2 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 2 device detected a negative direction, that is, decreasing counts. 2…255 = reserved The value for Hookup Test Feedback 2 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect the axis reaching the configured Motor Encoder Test Increment within 2 seconds, the servo sets the test voltage back to zero and disables the drive. The control reflects this condition through the Test Status axis output parameter. This usually indicates that either the cabling to the drive or the cabling to the encoder is incorrect. Running MRHD with the Encoder Hookup Test selected is an effective method of isolating the problem to the encoder or drive.

Encoder Hookup Test

If the Encoder Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects that the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test is complete. The motion module then reports the direction of travel as one of the following MRHD output parameters.

Table 124 - MRHD Output Parameters

Axis Parameter	Data Type	Units	Definition
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions can occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7…255 = reserved
Hookup Test Feedback Direction 1	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 1 device. 0 = The drive's feedback 1 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 1 device detected a negative direction, that is, decreasing counts. 2…255 = reserved The value for Hookup Test Feedback 1 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

Table 124 - MRHD Output Parameters

Axis Parameter	Data Type	Units	Definition
Hookup Test Feedback Direction 2	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 2 device. 0 = The drive's feedback 2 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 2 device detected a negative direction, that is, decreasing counts. 2…255 = reserved The value for Hookup Test Feedback 2 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect the axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test by using the MAS instruction and check the encoder wiring.

Marker Hookup Test

If the Marker Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects a marker (Channel Z) pulse, the test is then complete. The motion module then reports success via the Test Status.

Table 125 - MRHD Test Status Parameters

Axis Parameter	Data Type	Units	Definition
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions can occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7…255 = reserved
Hookup Test Feedback Direction 1	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 1 device. 0 = The drive's feedback 1 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 1 device detected a negative direction, that is, decreasing counts. 2…255 = reserved The value for Hookup Test Feedback 1 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.
Hookup Test Feedback Direction 2	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 2 device. 0 = The drive's feedback 2 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 2 device detected a negative direction, that is, decreasing counts. 2…255 = reserved The value for Hookup Test Feedback 2 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect that axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test by using the MAS instruction and check the encoder wiring.

Commutation Test

The Commutation test only applies to PM motors. This test applies current to the motor to align the rotor and check for proper phasing of a UVW encoder or Hall sensor, if applicable. Finally, the test measures the commutation offset.

Table 126 - MRHD Commutation Test Parameters

Axis Parameter	Data Type	Units	Definition
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions can occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7…255 = reserved
Hookup Test Commutation Polarity	USINT	-	Reports if the UVW phasing of the Encoder or Hall Sensor match the phasing of the Motor. If the motor and UVW commutation phasing do not match the Commutation Polarity is Normal. If it is determined that the phasing for the motor and commutation device do not match, this parameter reports that the Commutation direction is Inverted. This value can be used to configure the Commutation Polarity attribute. 0 = normal 1 = inverted 2…255 = reserved
Hookup Test Commutation Offset	Real	Electrical degrees	The Hookup Test Commutation Offset reports the measured commutations offset of a PM motor during the Commutation Test. This represents the value that must be applied to the motor position accumulator in order to align the Electrical Angle signal with motor stator windings. This value can be used to configure the Commutation Offset attribute.

Test Status

This parameter returns the status of the last Run Hookup Test service on the targeted drive axis. Conditions can occur that make it impossible for the control to properly perform the test operation. When this is the case, the test process is automatically aborted and a test fault is reported and stored in the Hookup Test Status output parameter. Possible values for Test Status are shown in this table.

Table 127 - Test Status Values

Error Message	Code	Definition
Test Success	0	Test process has been successful.
Test In Process	1	Test is in progress.
Test Aborted	2	Test Process was aborted by user.

Table 127 - Test Status Values

Error Message	Code	Definition
Test Time-out	3	Test Process has exceeded timed out (2 Seconds)
Test Servo Fault	4	Test Process Failed due to Servo Fault
No Feedback 1	5	Test Process Failed - no feedback 1 counts
No Feedback 2	6	Test Process Failed - no feedback 2 counts

Executing the Instruction

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

  For more information, see Structured Text Programming on page 359.

Arithmetic Status Flags

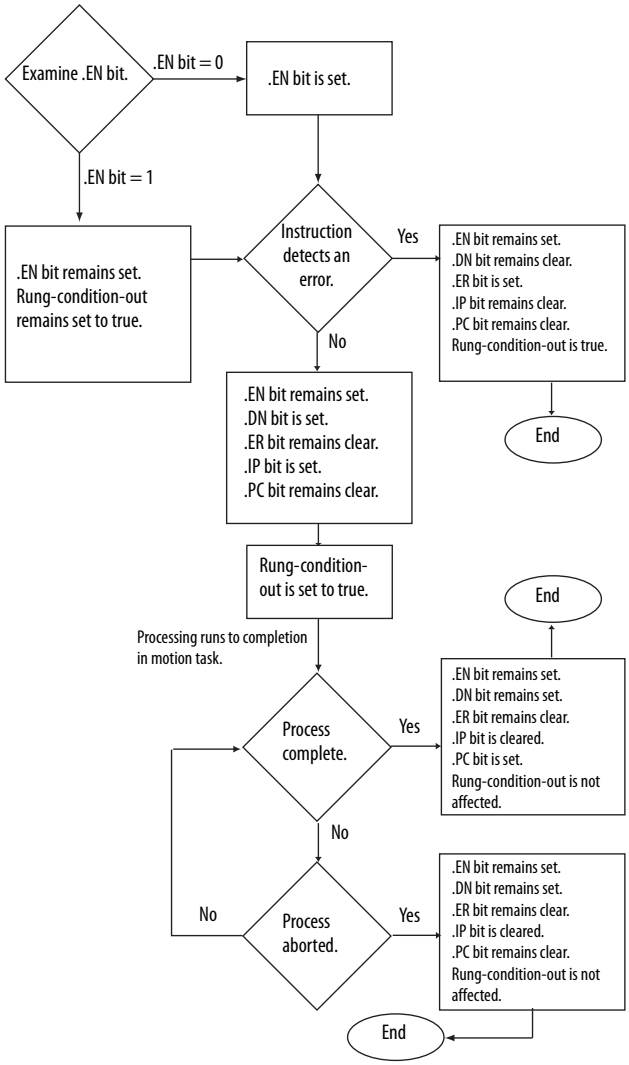
The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

MRHD Execution Conditions

Condition	Ladder Diagram Action	Structured Text Action
Prescan	The .EN, .DN, .ER, .IP, and .PC bits are cleared. The rung-condition-out is set to false.	No action taken.
Rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN, .ER, .IP, and .PC bits are not affected. The rung-condition-out is set to false.	N/A
Rung-condition-in is true	The instruction executes. The rung-condition-out is set to true.	N/A
EnableIn	N/A	EnableIn is always set. The instruction executes.
Instruction execution	See the following figure.	
Postscan	The rung-condition-out is set to false.	No action taken.



Error Codes

See Error Codes (ERR) for Motion Instructions on page 345.

MRHD Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MRHD instruction receives a Servo Message Failure (12) error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Process terminated on request (1)	Test execution followed by an instruction to shutdown/disable drive, or a motion stop instruction or a Processor change requests a cancel of the Test.
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Incorrect Tune Process order. (sercos)

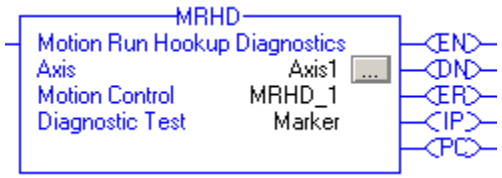
MRHD Changes to Status Bits

Bit	State	Meaning
DriveEnableStatus	TRUE	• The axis is in the drive control state. • The drive enable output is active while the tuning profile is running.
TestStatus	TRUE	The axis is running a testing process.

MRHD Example

When the input conditions are true, the controller runs the *encoder* diagnostic test on *axis1*.

Relay Ladder



Structured Text

```
MRHD(Axis1,MRHD_1,Marker);
```

MDSC Functionality

The Master Driven Speed Control (MDSC) function provides the ability to synchronize one or more motion axes to a common Master Axis. The MDSC uses the Motion Master Driven Axis Control (MDAC) instruction, which assigns a Master:Slave relationship for single axes.

The MDSC function provides synchronization of axes without the use of CAM profiles. The MDSC functions can also be used with Time Driven CAM profiles (MATC) to make them act like position driven profiles synchronized with a Master Axis.

Master Driven Axis Control (MDAC)

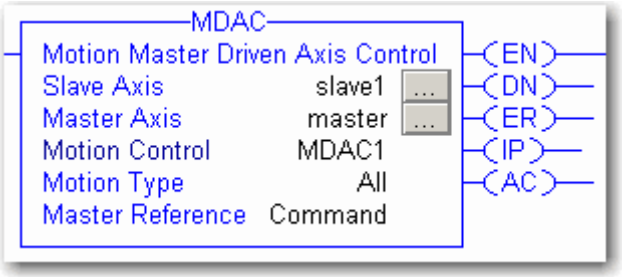
The Master Driven Axis Control (MDAC) instruction assigns a Master:Slave relationship for a single axes.

Operands

The MDAC instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder



Structured Text

```
MDAC (slave1, master, MDAC1, All, Command);
```

Table 128 describes the Relay Ladder operands.

Table 128 - MDAC Operand Descriptions

Operand	Type	Format	Description
Slave Axis	AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_CIP_DRIVE AXIS_VIRTUAL	Tag	The single axis being controlled by the Master Axis when the motion planner is in Master Driven mode. The Slave Axis can be any axis that has been configured.
Master Axis	AXIS_CONSUMED AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_CIP_DRIVE AXIS_VIRTUAL	Tag	Any configured single axis that the Slave Axis follows. The Master Axis can be any axis that has been configured.
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.
Motion Type	UNIT	Immediate Tag	Specifies the move type (MAM, MAJ, MATC, or MAM - Master Offset Move for a Position Cam) executing on the Slave Axis that will be controlled by the Master Axis when a single axis motion instruction is programmed in Master Driven Mode. 0 = All 1 - Move 2 = Jog 3 = Time Cam 4 = Master Offset Move
Master Reference	UNIT	Immediate Tag	Selects the Master Axis position source as either Actual Position (0) or Command Position (1).

Table 129 describes the Ladder and Structure text.

Table 129 - Example MDAC Instruction Parameter Descriptions

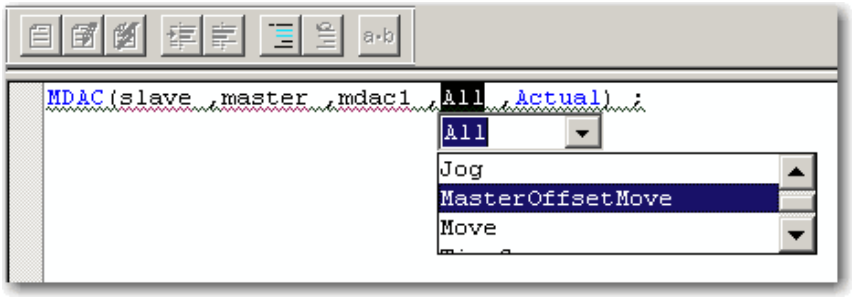
Parameter	Description
Slave1	Slave1 is the axis that is being controlled by the Master Axis when the motion planner is in Master Driven mode.
Master	Master is the axis that slave1 axis will follow.
MDAC1	MDAC1 is the control tag for the MDAC instruction.
All	The Master Axis will control all the single axis move types, for example, MAM, MAJ and MATC, that are executed on the Slave Axis when a single axis motion instruction is programmed in Master Driven mode.
Command	Indicates that the Commanded Position of the Master axis is used as the position reference.

Structured Text

MDAC (SlaveAxis, MasterAxis, MotionControl, MotionType, MasterReference);MDAC(Group,MotionControl)

The operands are the same as those for the relay ladder MDAC instruction.

You have the option to browse for enumerations in the Structured Text Editor.



MOTION_INSTRUCTION

The MDAC instruction is used to select a single axis as a Master Axis and a single axis as a Slave Axis. The MDAC instruction connects a Master and Slave Axis for a MAM, MAJ, and MATC instruction. When an MDAC is executed (goes IP), the specified Slave Axis in the MDAC instruction is logically geared with the designated Master Axis. After motion on both the Master and Slave Axes has been initiated, the Slave follows the Master Axis motion by using the programmed dynamics of the motion instruction.

Table 130 describes the bit leg definitions for the MDAC instruction.

Table 130 - MDAC MOTION_INSTRUCTION Bit Leg Descriptions

Bit	Enumerations	Description
31	.EN (Enable)	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
29	.DN (Done)	The done bit is set when the Master Driven Axis Control instruction is successfully initiated.
28	.ER (Error)	The error bit is set when there is an invalid combination of parameters in the MDAC instruction.
26	.IP (In Process)	The in process bit is set when the MDAC instruction is activated and reset by an instruction (for example, the MASD instruction).
23	.AC (Active)	The active bit is set when a move (MAJ, MAM or MATC) goes IP in Master Driven mode on the axis that is selected as the Slave Axis of the MDAC instruction. The AC bit will be reset when all single axis motion that is being controlled by the MDAC is completed. If the Slave Axis is started in Time Driven mode, then the AC bit of the MDAC does not go active. The IP bit of the MDAC instruction does not change at this time.

There are no changes in any active motion when a new MDAC instruction is activated. Activating a new MDAC instruction puts the parameters programmed in the MDAC into a pending state (IP bit is set on the MDAC). The parameters in the pending MDAC instruction are overridden if you execute a succeeding MDAC before a new single axis motion instruction (MAM, MAJ, and MATC) is activated on the slave axis. The IP bit of the old pending MDAC is cleared and the IP of the new pending MDAC is set. The most recent values in the pending MDAC instruction are used when a new single axis motion instruction is activated on the Slave Axis.

Interaction of MDAC with Time Cams (MATC)

A MATC with an Execution Schedule of Pending always uses the Instruction mode of the active MATC; Lock Position and Lock Direction of the pending MATC are ignored. This is explained further in the following example.

Assume that there are four instructions executed in the following order:

Table 131 - Example MDAC with Time CAMS (MATC)

Parameter	Description
MDAC1	Master=X, Slave=Y
MATC1	Slave Y, Immediate
MDAC2	Master=Z Slave=Y MDAC2 is executed (goes IP) while MATC1 is moving (Master Axis is changed)
MATC2	Slave=Y, Pending

MATC2 will use the parameters from MDAC1 (Master and Slave) because MATC1 was moving (active) when MDAC2 was executed. If MATC2 and MDAC2 were executed in the reverse order, you get the same results. This follows the existing paradigm of MAPC instruction where the pending CAM uses the same master as the running CAM.

Motion Control

The Motion Type and Master Reference bits are affected by the MDAC instruction.

Table 132 - Bits Affected by the MDAC Instruction

Operand	MOTION_INSTRUCTION	Tag	Value
Motion Type	UNIT32	Immediate Tag	Enumerations: 0 = All 1 = Move 2 = Jog 3 = Time Cam = 3 4 = Master Offset Move
Motion Type specifies the move type executing on the Slave Axis. Move types include, MAM, MAJ, MATC, MAM, or Master Offset Move for a Position Cam. The Slave Axis that will be controlled by the Master Axis, specified in the MDAC, when a single axis motion instruction is programmed in Master Driven mode.			
Master Reference	UNIT32	Immediate Tag	0 = Actual Position 1 = Command Position
Master Reference selects the Actual Position (0) or Command Position (1) of the master axis that will be used to control the salve axis in Master Driven Mode.			

When the Motion type is set to All then the Master Driven mode applies to all single axis instructions, that is, MAM, MAJ, MATC, and a MAM (Master Offset Move). A runtime error is caused when switching from All to any other motion type and vice versa. You must disable MDAC before performing switch from All to any other type and vice versa.

You can assign the same Slave axis to two different Master axes, however, a different motion type must be used for each axes. For example, both X and Y can not be the master for the MAM motion type of Z. If X and Y are both the Master axes and Z is a Slave axis, then X can be assigned to be the Master axis for MAM instructions for Z. Y can be assigned to be the Master axis for MAJ instructions for Slave Z.

Table 133 - Assign the same Slave to Different Masters

Instruction	Description
MDAC, Move=MAM, Master=X Slave=Y	assigns master and slave axes
MDAC, Move=ALL, Master=Z Slave=Y (or any other axis)	causes an error because the previous MDAC assigned the MAM to the Y axis.

The MDAC mode becomes **enabled** once the MDAC instruction is executed. The MDAC mode becomes **active** after being enabled and a motion instruction with the MDAC speed unit selection is executed.

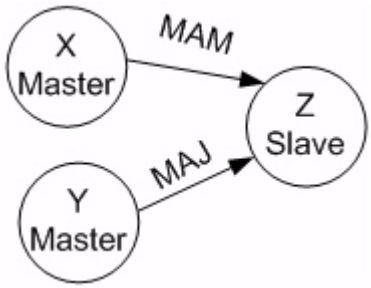
If there is an active MDAC (All) on a Slave, then setting the MDAC instruction to a Motion Type other than All causes a runtime error.

To reassign the Master once anything other than the All option is used, you must execute any of the following:

- On the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, or MGSD
- On the Master Axis: MASD, MCSD, or MGSD

If you assign the same axis to be both a Master and a Slave Axis, an RSLogix 5000 software verification error is generated.

Figure 35 - Master and Slave Relationship



You can change the Master Axis for an active Slave, which is in motion by executing a MDAC for the Slave with a different Master.

However, If the same motion type is assigned to two different masters an error will result, for example, both X and Y can not be the master for the MAM motion type of Z then the new MDAC instruction is put in a queue for the slave and will become active the next time a motion instruction with the MDSC unit selection with this master becomes active in either Replacement or Merge mode.

Table 134 - Switching Master Axes

Instruction	Description
MDAC Motion Type=MAM Master=X Slave=Z	Assigns master and slave axes.
MAM X	Start MAM motion on master 1.
MAM Z	Start MAM motion on the slave using master 1.
MAM Y	Start MAM motion on the Y axis that is on the future master 2.
MDAC Motion Type=MAM Master=Y Slave=Z	Assign Y to be the future pending Master axis.
MAM Z	Start new MAM motion on the slave Y will be the master for the new motion.

EXAMPLE

The MDAC mode becomes **enabled** once the MDAC instruction is executed. The MDAC mode becomes **active** after being enabled and a motion instruction with the MDAC speed unit selection is executed.

If there is an active MDAC (All) on a Slave, then setting the MDAC instruction to a Motion Type other than All causes a runtime error.

To reassign the Master once anything other than the All option is used, you must execute any of the following:

- On the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, or MGSD
- On the Master Axis: MASD, MCSD, or MGSD

If you assign the same axis to be both a Master and a Slave Axis, an RSLogix 5000 software verification error is generated.

You must use different control words on all MDAC instructions. If the same control word is used for the active and pending MDAC, the IP bit of the pending MDAC will not work properly. No error detection is made if the same control word is used on multiple MDAC instructions.

When separate axis control words are used for the active and the pending MDAC instructions, the active and the pending MDAC will both have their IP bits set. However, the active MDAC will only have its AC bit set. This is standard operation for all instructions.

Master Reference

The Master Reference for an MDAC instruction selects the Master Axis position source. The enumerations for Master Reference Axis are Actual and Command:

- Actual – Slave motion is generated by using the actual (current) position of the Master Axis as measured by its encoder or other feedback device.
- Command – Slave motion is generated by using the command (desired) position of the Master Axis.

  Because there is no Command Position for a Feedback Only Axis, if you select either Actual or Command for Master Reference of a Feedback Only axis, the Actual Position of the Master Axis is used. The Actual and Command Position are always the same for this case. No error is generated.

  Because there is no Actual Position for a Virtual axis, if you select either Actual or Command for Master Reference, the Command Position is used. No error will be generated.

An error is generated if a MDAC instruction is executed that changes the Master Reference of a Slave Axis that is in motion. If the new MDAC instruction errors for any reason, the original MDAC will remain active.

Motion Direct Command and the MDAC Instruction

There is no Motion Direct command for MDAC. To obtain Motion Direct command support for Master Driven Speed Control mode, you must first program an MDAC before you execute an MAM or MAJ in Master Driven mode as a motion direct command.

Arithmetic Status Flags

Not affected.

Fault Conditions

See Actions Taken When Stopping/Shutdown Instructions are Executed on the Slave Axis on page 307 for the fault descriptions for motion instructions when MDAC is active.

Error Codes

See Motion Error Codes (.ERR) in Appendix AError Codes (ERR) for Motion Instructions on page 345 for run time errors for motion instructions when MDAC is active.

Logix Designer Application Verification Errors

An invalid or No Master Axis causes errors to be generated when verified by the Logix Designer application. The following conditions can cause this error:

- The Master and Slave Axis are the same.
- Master or Slave Axis is not configured.
- Master or Slave Axis is inhibited.
- Redefine position is in progress.
- Home of an axis is in progress.
- MDSC ALL Conflict. The ALL parameter is not allowed while a Slave axis motion generator, for example, jog motion, is already assigned.

Master Driven and Time Driven Modes

Changing the motion mode between Master Driven and Time Driven mode and vice versa is automatically performed when a new motion instruction (such as, MAM, MAJ, and MATC) is activated if the new instruction has been programmed in a different mode than the active motion instruction.

When the new motion instruction is activated, the system will assume that the desired mode for the new instruction is the mode (Master Driven or Time Driven) as specified in the programmed units of the speed parameter contained in the new instruction. At all times, including when changing modes, the Accel, Decel, and Jerk must all be programmed in the same units as the Speed parameter or the instruction will get a MDSC_UNITS_CONFLICT_ERROR error.

A runtime MDSC_INVALID_MODE_OR_MASTER_CHANGE error will occur only if you attempt to change from Time Driven mode to MDSC mode, or vice versa with an MCD instruction.

If you change from Time Driven mode to Master Driven mode while an axis is moving and Lock Direction is not Immediate Forward or Reverse you will get error 95, DSC Lock Direction Conflict.

If both the master and slave axes are idle (for example, paused), the MAM or MAJ can make a change on the slave. However, the error MDSC IDLE_MASTER_AND_SLAVE_MOVING is generated if MDSC mode is started while the slave is moving when the master is idle.

Different Time Driven and Master Driven modes can be used for different motion types for superimposed motion. For example, the MAM can be in Time Drive mode and the MAJ can be in Master Driven mode for the same Slave Axis.

Changing the Master Axis

The following sequence of events must be followed to transfer a Slave Axis from one Master Axis to a second Master Axis:

- First, you must execute an MDAC instruction to reassign the Slave Axis from the first Master Axis to the second Master Axis. This makes the reassignment pending. The IP bit of the pending MDAC instruction is set as an indication of the pending reassignment.
- Second, you must execute a new motion command (for example, an MAM or MAJ). The axis becomes unlocked from the first Master Axis and reassigned to the second Master Axis when this new motion instruction is executed (goes IP).

EXAMPLE

The MDAC mode becomes **enabled** once the MDAC instruction is executed. The MDAC mode becomes **active** after being enabled and a motion instruction with the MDAC speed unit selection is executed.

If there is an active MDAC (All) on a Slave, then setting the MDAC instruction to a Motion Type other than All causes a runtime error.

To reassign the Master once anything other than the All option is used, you must execute any of the following:

- On the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, or MGSD
- On the Master Axis: MASD, MCSD, or MGSD

If you assign the same axis to be both a Master and a Slave Axis, an RSLogix 5000 software verification error is generated.

If the slave is not moving when the master axis is changed then no problem will ever occur. However, if the slave is in motion when the change in master axes occurs then the final effective slave speed is computed as the product of the Master Axis' speed and the slave's programmed speed. If the new final effective Slave Axis speed is less than 5% of its original speed after a change in the Master Axis, then the change will not be allowed and the MDSC_INVALID_SLAVE_SPEED_REDUCTION error will occur. This will always be the case if the second master axis is idle (velocity=0). In this case, the motion instruction making this request receives an MDSC_IDLE_MASTER_AND_SLAVE_MOVING error.

If the second Master Axis is moving while the change in the master axis is being made, look at the TrackingMaster instruction status bit of the Motion Control words. The Motion Control words of the motion instruction that is performing the change displays when the change in masters is finished.

There are two reasons why the slave might accelerate. One, when the master driving the slave is accelerating and two, when the slave is accelerating due to it's programmed acceleration in a new instruction.

When the acceleration of the Tracking Master bit in the instruction Control Word is cleared, it is not sensitive to the acceleration due to the master being accelerated. The master is only sensitive to the acceleration of the slave caused by a programmed change in the speed of two successive instructions.

Actions Taken When Stopping/Shutdown Instructions are Executed on the Slave Axis

All commands in the following table are for the Slave Axis system. The following table identifies the change in state of the MDAC link between the master and slave axis as each instruction in column 1 is executed on the slave axis.

Table 135 - Actions for the Slave Axis

Instruction	Parameters	MDAC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Reset
MCSD		Reset
MAS	Stop Type = Jog	Not Changed AC bit for the MDAC is set because a MAJ is active, then the AC bit will be reset.
	Stop Type = Move	Not Changed AC bit for the MDAC is set because a MAM is active, then the AC bit will be reset.
	Stop Type = Time CAM	Not Changed AC bit for the MDAC is set because a MATC is active, then the AC bit will be reset.
	Stop Type = All	Reset AC bit for the MDAC is set because any single axis motion command is active then the AC bit will be reset.
MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Reset
	Disable DRV	Reset
	Shutdown	Reset

If the same Slave Axis or a Slave Coordinate System is controlled by multiple Master Axes. If one MDAC or MCCD relationship that contains the Slave or Slave Coordinate System is broken, then all MDAC or MDCC relationships that contain the Slave or Slave Coordinate System will be broken.

The MDAC link is broken when the following instructions are executed on the slave axis:

- MAS (All), MCS (All), MGS, MASD, MCSD, MGSD, a mode change.
- The MAS (anything other than All) and MCS (coordinated) do not break the MDAC link.

A mode change (Rem Run to Rem Prog or Rem Prog to Rem Run) or an axis fault also breaks the MDAC link.

For information on the MDCC instruction and the other coordinate instructions, see the Coordinate System User Manual, publication MOTION-UM002.

Actions Taken When Stopping/Shutdown Instructions are Executed on the Master Axis

The following table identifies the change in state of the MDAC link between the master and slave axis as each instruction in column 1 is executed on the master axis. All commands in the following table are for the master axis.

Table 136 - Actions for the Master Axis

Instruction	Parameters	MDAC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Not Changed
MCSD		Reset
MAS	Any Stop Type Jog Move Time CAM All	Not Changed
MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Not Changed
	Disable DRV	Not Changed
	Shutdown	Reset

If the same Master Axis is controlling multiple Slaves or a Slave Coordinate System, then all MDAC or MDCC relationships that contain the Master Axis are broken.

The MDAC link is broken on the Master Axis when the following instructions are executed on the Master Axis:

- MASD, MCSD, and MGSD. The MAS and MCS instructions for any Stop Type, including All, do not break the MDAC link.
- A mode change (Rem Run to Rem Prog or Rem Prog to Rem Run) or an axis fault also breaks the MDAC link.

Master Driven Speed Control Parameters for Single Axis Motion Instructions

Before any of the MDSC parameters, identified in Table 137, are used by the RSLogix 5000 instructions MAM, MAJ or MATC, you must execute an MDAC instruction containing the axis as a slave and it must be active. This table identifies which parameter is applicable to the following single axis motion instructions, that is, to MAM, MAJ or MATC and the mode it is applicable in.

Table 137 - MDSC (Master Driven Speed Control) Parameter Structure

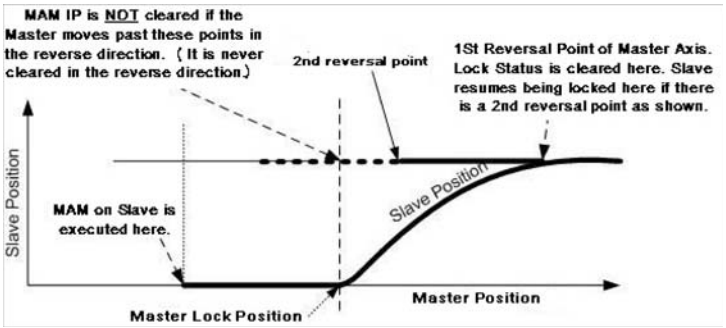
Parameter	Instruction	Mode
Input Parameters		
Lock Direction	MAM, MAJ, MATC	Master Driven Only
Lock Position	MAM, MAJ, MATC	Master Driven Only
Event Distance	MAM	All modes (Master Driven or Time Driven)
Instruction Mode	MATC	Identifies the mode of the instruction to be either Time Driven (0) or Master Driven mode (1)
Output Parameter		
Calculated Data	MAM	All modes (Master Driven, Time Driven, and Timed Based)

Table 138 - MDSC Lock Direction Parameter Description

Input Parameter	Data Type	Description	Value
Lock Direction	Immediate	This parameter is used for both Time Driven and Master Driven mode. In master driven mode the Lock Direction is used by the axis specified as the Master Axis in the MDAC instruction. It is not used in Time Driven mode. The first word of the Lock Direction (Immediate or Position) enumeration definition identifies the lock type as either: • Immediate (lock is performed immediately) • Position (lock is performed when the Master Axis crosses the specified Lock Position) The second word of the Lock Direction (Forward or Reverse) specifies the direction in which the Master Axis has to be moving when it crosses the Lock Position for the lock to take effect. Forward is positive velocity, reverse is negative velocity. The instruction will error with MDSC_LOCKDIR_CONFLICT (95). The enumeration NONE must be used in time driven mode or the instruction will error. On of the other 4 enumerations must be used in master driven mode. For an MAM instruction, the Slave always moves in one direction, it's programmed direction. Once it starts moving it follows the Master Axis in the programmed direction, regardless of the direction of the Master Axis. If the Master reverses, the slave stops. These are the Master Driven mode enumerations.	Default = 0 0 = None 1 = Immediate Forward Only 2 = Immediate Reverse Only 3 = Position Forward Only 4 = Position Reverse Only

Table 139 - Enumeration Descriptions for the MDAC Input Parameter

Enumeration	Name	Description
0	None	Indicates that the Lock Position is not active. If Lock Direction is set to None and the Master Driven mode is selected by the speed parameter of the motion instruction, a run time error is generated. Conversely, if Lock direction is set to a value other than None and the speed parameter units indicate Time Driven mode, an error is also generated.
1	Immediate Forward Only	Motion starts immediately when the Master is moving in the Forward Direction. The Master Axis is followed only while it is moving in the Forward Direction.
2	Immediate Reverse Only	Motion starts immediately when the Master Axis is moving in the Reverse Direction. The Master Axis is followed only while it is moving in the Reverse Direction.
3	Position Forward Only	Motion starts, for example, the Slave locks to the Master Axis when the Master Axis crosses the Lock Position while it is moving in the Forward Direction. The Master Axis is followed only while it is moving in the Forward Direction. If the start position of the Master Axis equals the Lock Position and this enumeration is selected, then motion will not start because the Lock Position will not be crossed.
4	Position Reverse Only	Motion starts when the Master Axis crosses the Lock Position while it is moving in the Reverse Direction. The Master Axis is followed only while it is moving in the Reverse Direction. If the start position of the Master Axis equals the Lock Position and this enumeration is selected, then motion will not start because the Lock Position will not be crossed.

Input Parameter	Data Type	Description	Value
Lock Position	IMMEDIATE REAL or TAG	**Lock Position in Master Driven Mode** After the slave axis motion has been initiated by a MAM or MATC, it will go IP but will not start moving until the master axis crosses the Master Lock Position. This is an absolute position (plus or minus) on the Master Axis in Master Axis units. You can specify a Lock Position to delay the start of motion of a Slave Axis after the motion instruction has been initiated on the Slave Axis. If a Slave Axis is already moving and a second move instruction of the same type (MAM, MAJ, MATC) is activated on the same Slave with a Lock Position, then you will receive an MDSC LOCK WHILE MOVING error for the second instruction. Because Merge is always performed immediately when an instruction is enabled, a merge instruction with both a Lock Position and a Merge enabled that is executed on a Slave Axis when it is at a nonzero velocity will receive an MDSC LOCK WHILE MOVING error. The Lock Direction determines the direction in which the Master Axis must be moving when it crosses the Lock Position before the Slave locks to the Master Axis. If there is an unwind value specified on the Master Axis, then the Master Lock Position must be between 0 and the unwind value (that is, the Lock Position cannot be more than one unwind.) This parameter is used only in Master Driven mode. **Lock Position in Time Driven Mode** In Time Driven mode, the Lock Direction must be set to None or an error will be generated. **Axis Lock Behavior** When the Master axis crosses the Master Lock position in the direction as specified by the motion instruction, the slave becomes locked to the Master axis. The LockStatus bit of the Slave axis status word is set at this time. The MAM and MAJ instructions on the slave axis in MDSC mode go IP as soon as they are processed by the motion planner. For the Immediate Forward Only or Immediate Reverse Only Lock Directions, the slave gets locked to the Master Axis immediately when the instruction is executed (goes IP). For the Position Forward Only or Position Reverse Only Lock Directions, the slave gets locked to the master axis when the master axis crosses the Master Lock Position in the direction as specified by the motion instruction. In either case, the LockStatus bit of the Slave position status word is set when the lock occurs. Because there is no bidirectional behavior defined, once locked, the Slave follows the Master only in the specified direction. If the Master reverses direction, then the Slave stops following the Master. The LockStatus bit remains set until the Master decelerates to zero prior to reversal. It is cleared at the point of reversal of the Master axis. The Slave does not follow the Master while the Master travels in the reverse direction. If the Master axis changes directions again, then the axis LockStatus bit is set again when the Slave Axis crosses the original reversal point at which time the slave resumes following the Master Axis. **Figure 36 - Axis Lock Behavior**  MDSC_LOCK_DIR_MASTER_DIR_MISSMATCH is generated if a new instruction with the Lock Direction parameter in the opposite direction of the current master direction is merged or replaces an active motion instruction (on the Slave Axis). For example, if the programmed Lock Direction is Immediate Forward Only and the master axis is moving in reverse direction. A new instruction used to merge an active instruction on the Slave Axis must use the Immediate Forward Only or Immediate Reverse Only Lock Direction. If the new instruction, which is merged, uses the Position Forward Only or Position Reverse Only Lock Direction, the error MDSC_LOCKDIR_CONFLICT is generated on the new motion instruction.	Default = 0

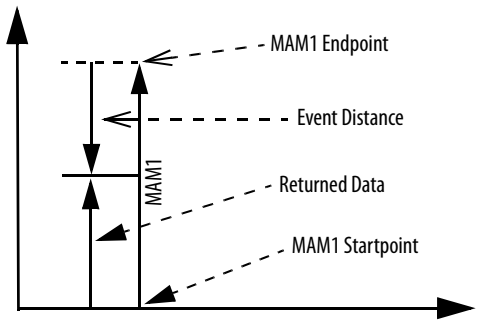
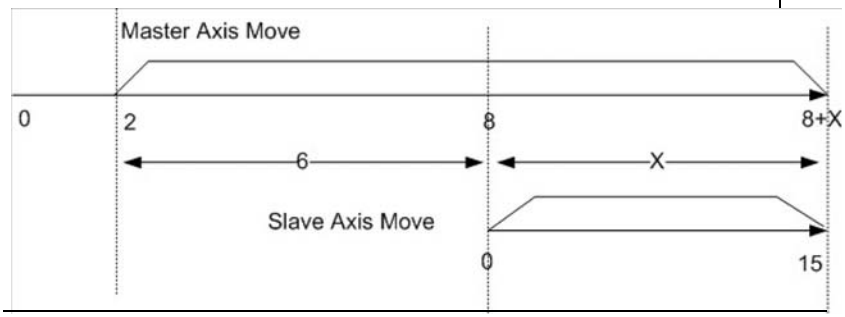
Input Parameter	Data Type	Description	Value
Event Distance	ARRAY	The position(s) on a move measured from the end of the move. This is an array of input values that specifies the incremental distances along the move on the Slave. Each member of the array is measured as follows: • Distances are measured starting from the end of the move towards the beginning of the move as shown in this figure. **Figure 37 - Event Distance Measured**  If the value in the Event Distance array is 0.0, then it is the time or distance for the whole move. If the value is greater than or equal to the move length then a 0 is returned. The values entered in the Event Distance array are the same for both Time Driven and Master Driven Mode, only the returned values in the Calculated Data array are different depending on the programmed mode of the slave axis. When Event Distance is specified as a negative number, then the Event Distance calculation is skipped and a -1 is returned in the Calculated Data array for the specified Event Distance parameter. There is no limit on the dimension of either the Event Distance or Calculated Data arrays. However, a maximum of 4 elements (the specified value and the next 3) of the Event Distance array will be processed. Special consideration for the rare case of an overshoot when a MCD or MCCD is done close to the moves endpoint. For this case, the Calculated Data will include the overshoot when the Event Distance is 0, because the master will have to traverse this amount for the move to finish. For other Event Distances, the overshoot will not be included.	Default = 0 No Event Distance array or a REAL array tag. The array must be a minimum size of 4. If the array is greater than 4, only the first four locations specified are used.
Instruction Mode	INIT32	Specifies if an MATC should be executed in Time Driven Mode (0) or Master Driven Mode (1).	Valid = 0 or 1 Default value = 0

Table 140 - Output Parameter Description

Output Parameter	Data Type	Description	Value
Calculated Data	REAL ARRAY or 0	The calculated output for the Event Distance input parameter, that is the Master Distance(s) (or time) measured from the beginning of the move to the Event Distance point. The returned Calculated Data value is dependent on the following: • The instruction type, (that is, MAM for single axis and MCLM/MCCM for coordinated motion) • The mode of the Slave Axis (that is, Time Driven or Master Driven). • If superimposed motion is active, the Calculated Data does not include any of the superimposed motion. The returned Calculated Date value is depended on the following modes. **Master Driven** The returned Calculated Data parameter is the incremental delta Master position that is needed to make the Slave Axis move from the point at which Slave is locked to the Master and starts moving along the programmed path, to the point where distance to go is less than the specified Event Distance. If the specified data in the Event Distance is array element is 0.0, then the master needed for the entire move to complete is returned. **Time Driven** The returned data in the Calculated Data parameter is the total time in seconds that is needed to make the axis move from the move's start point to a point where distance to go is less than the specified Event Distance. If the specified data, in the Event Distance is array element is 0.0, then the time it takes the entire move to complete is returned. If the value is greater than or equal to the move length then a 0 is returned. The Logix Designer application Motion Planner processes and calculates output data and places the result in the Calculated Data array as supplied in the instruction. The number of calculated array elements stored in the Calculated Data array is based on the follow conditions: • The number of elements in the Event Distance array. • For each of the first 4 elements Event Data array, one element will be computed and placed in the Calculated Data array. • The fifth element and beyond of the Event Distance array are ignored. Existing values in the Calculated Data array are overlaid when the Event Distance array is processed. A -1 will be returned in the Calculated Data array for each negative value in the Event Distance array. No Event Distance calculation is made for these array elements. You can change the Event Distance array elements dynamically in the program. However, if the Event Distance is changed after the instruction has been initiated (that is, the IP bit has been set), then the change is ignored. An error is generated if size of the Calculated Data array is smaller than the Event Distance array. The default value for versions when bringing old systems forward (earlier than v20) is 0, signifying that there is no Event Distance array. If the Event Distance is greater than the move length, internally it will be forced to equal the move length. **Example 1** Event Distance array = [11, 22, -5, 23, 44] Calculated Data array = [f(11), f(22), -1 ,f(23)] Where f is the calculated data function.	Default = 0 No calculated Data array or a REAL array tag

Table 140 - Output Parameter Description

Output Parameter	Data Type	Description	Value
		The 44 is ignored because it is the fifth element in the Event Distance array; a -1 is returned in the third element of the Calculated Data array because the corresponding Event data Array element is negative. A status bit (CalculatedDataAvailable) in the existing motion instruction status word has been defined to indicate that the requested data in the Event Distance has been returned in Calculated Data. Only one status bit is used to indicate all Calculated Data is available. The Calculated Data is set only once in the instruction queue or planning process. It is not updated as the move occurs to reflect distance to go. It is updated for a change dynamics, however. Assume that the master axis is at a position of 2.0. The slave is programmed to an incremental value of 15.0 with a Master Lock Position at 8.0. The Event Distance is set to 0.0, which means that we want the total Master Distance (X in Table 38) needed for the slave to move 15.0 units starting when the Master is locked at a position at 8.0. The incremental value of X is returned in the Calculated Data parameter. **Figure 38 - Master Distance X** 	

Speed Enumerations

Common enumerations are used for the speed parameter of all motion instructions. Some instructions accept only a limited subset of the speed enumerations. Checks for valid unit combinations are done at instruction execution time. Some enumerations that are in the following table are not used now but are reserved for future enhancements.

Additional tables are given in this chapter that further clarify which combinations are accepted in MDSC mode and which are accepted in Time Driven mode.

Table 141 - Speed Unit Parameter Descriptions

Mode	Enumerations	Compatibility
Time Driven	0 = Units per sec^2	Existing Enumeration
	1 =% Maximum	Existing Enumeration
	2 = Reserved	
	3 = Seconds Time based programming	New Enumeration

Table 141 - Speed Unit Parameter Descriptions

Mode	Enumerations	Compatibility
MDSC	4 = Units per MasterUnit	New Enumeration
	5 = Reserved	
	6 = Reserved	
	7 = Master Units Analogous to seconds in time based programming	New Enumeration

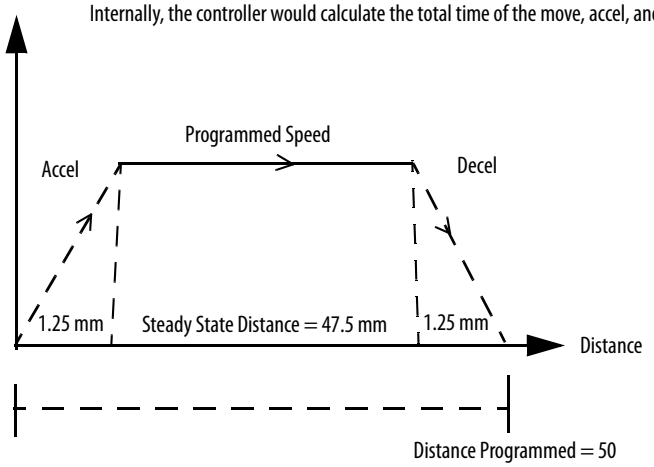
These rules must be followed to program the dynamics units (Speed, Accel/Decel, and Jerk) of all motion instructions.

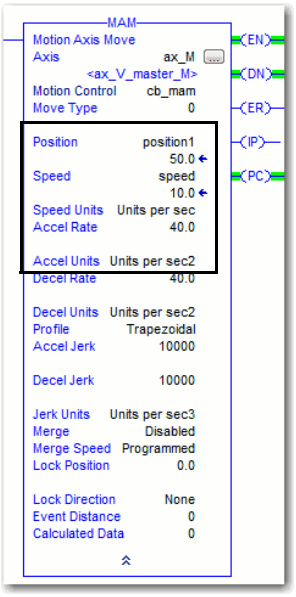
- When Speed is in either units/sec, %max, or seconds, then the instruction is considered to be in Time Driven mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- When Speed is in either Master Units or in Units/MasterUnit, then the instruction is considered to be in Master Driven mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode (Time Driven or Master Driven) or you get a runtime error.
- When speed is specified in time unit (seconds), the specified time is the total time of the move, including acceleration and deceleration time.
- When speed is specified in Master distance units, the specified distance is the total master distance of the move, including acceleration and deceleration distance of the Master Axis.

This figure is an example of speed as programmed in Logix Designer application, version 19 and earlier. You only had one option to program, speed directly as a rate in units of distance/time.

Figure 39 - Programming Rate in Logix Designer Application Version 19 and Earlier

In Logix Designer application, version 19 and earlier, you could only program speed as units. Internally, the controller would calculate the total time of the move, accel, and decel times.





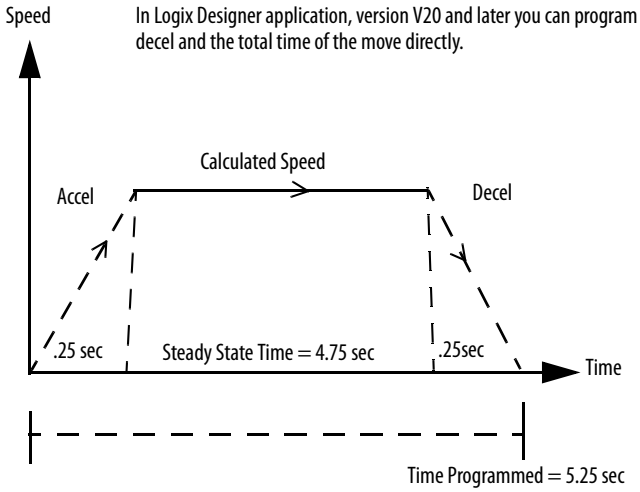
Logix Designer application version 19 and earlier MAM instruction programmed as rate.

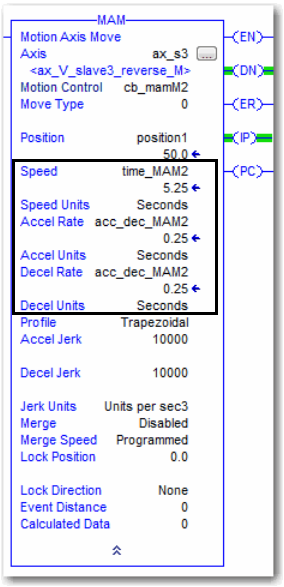
Position	50.0 mm (start 0.0)
Speed	10.0 mm/sec
Accel	40.0 mm/sec^2
Decel	40.0 mm/sec^2

So Travel_Distance = area under the curve [accel + at_speed + decel]
Travel_Distance = 50 mm
Travel_Distance = 50 mm [1.25 mm + 47.5 mm + 1.25 mm

In this figure, we are programming time. In Logix Designer application, version V20, the controller calculates the speed of the move: Speed & Accel/Decel as time [seconds].

Figure 40 - Programming Time in Logix Designer Application Version 20 and Later


Speed
In Logix Designer application, version V20 and later you can program the accel and decel and the total time of the move directly.
Calculated Speed
Accel
Decel
.25 sec
Steady State Time = 4.75 sec
.25sec
Time
Time Programmed = 5.25 sec



Logix Designer application version 20 and later MAM instruction programmed as time.

Position	50.0 mm (start 0.0)
Speed	5.25 sec
Accel	0.25 sec
Decel	0.25 sec

So Travel_Distance = area under the curve [accel + at_speed + decel]
Travel_Distance = 50 mm
Travel_Time = 5.25 sec [0.25 + 4.75 + 0.25 sec]

Acceleration and Deceleration Enumerations

The following enumerations are defined for Acceleration and Deceleration Unit parameters for motion instructions.

Table 142 - Acceleration and Deceleration Unit Parameter Descriptions

Mode	Enumerations	Compatibility
Time	0 = Units per sec^2	Existing Enumeration
	1 = % Maximum	Existing Enumeration
	2 = Reserved	
	3 = Seconds Time based programming	New Enumeration
MDSC	4 = Units per MasterUnit^2	New Enumeration
	5 = Reserved	
	6 = Reserved	
	7 = Master Units Analogous to seconds in time based programming	New Enumeration

The following table shows acceptable combinations of Speed, Acceleration, and Deceleration units.

Table 143 - Combinations of Speed, Acceleration, and Deceleration Units

Speed Units	Acceleration and Deceleration Units				
	Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Units per sec (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented	Not allowed - Time and Master Driven Units can not be combined.	
% Maximum (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented		
Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration		
Units per MasterUnits (Master Driven Mode Units)	Not allowed - Time and Master Driven Units can not be combined.			New Enumeration	Not Implemented
Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration

These rules must be followed to determine allowable Time and Master Driven mode when programming Acceleration and Deceleration units:

- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode or you get an error.
- If Speed units are Seconds, then Acceleration, Deceleration, and Jerk units must be seconds too.
- If Speed units are Master units, then Acceleration, Deceleration, and Jerk units must be Master units too.
- All unsupported unit combinations result in an err at runtime when the instruction is executed.

Jerk Enumerations

The following enumerations are defined for time driven and MDSC driven Jerk Units.

Table 144 - Time Driven and MDSC Driven Jerk Units Descriptions

Mode	Compatibility	Enumerations
Time	Existing Enumeration	0 = Units per sec^3
	Existing Enumeration	1 =% Maximum
	Existing Enumeration	2 = % of Time
	New Enumeration	3 = Seconds Time based programming
MDSC	New Enumeration	4 = Units per $MasterUnit^2$
		5 = Reserved
	New Enumeration	6 = % of Time-master Driven
	New Enumeration	7 = Master Units Analogous to seconds in time based programming

Acceptable combinations of Accel and Decel Units are based on the programmed Speed Units in the instruction as is shown in the following table. This table is used to clarify the differences in the following four tables.

Speed Units	Accel Units versus Jerk Units
Units per Sec	Table 145
Units per Master Units	Table 146
Seconds	Table 147
Master Units	Table 148

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Second.

Table 145 - Acceleration Units and Jerk Units when Speed Units are Units per Second

Jerk Units	Acceleration and Deceleration Units				
	Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Units per sec^3 (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented	Incompatible combinations of Time and Master Driven mode. A runtime error occurs.	
% of Time (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented		
Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented		
Units per MasterUnits^3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
% of Time-Master Driven (Master Driven Mode Units)					
Master Units (Master Driven Mode Units)					

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Master Unit.

Table 146 - Acceleration Units and Jerk Units when Speed Units are Units per Master Unit

Jerk Units	Acceleration and Deceleration Units				
	Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Units per sec^3 (Time Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. A runtime error occurs.	
% Maximum (Time Driven Mode Units)					
% of Time (Time Driven Mode Units)					
Seconds (Time Driven Mode Units)					
Units per MasterUnits^3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			New Enumeration	Not Implemented
% of Time-Master Driven (Master Driven Mode Units)				New Enumeration	Not Implemented
Master Units (Master Driven Mode Units)				Not Implemented	Not Implemented

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are in Seconds.

Table 147 - Acceleration Units and Jerk Units when Speed Units are in Seconds

Speed Units in Seconds		Acceleration (Speed in Seconds)				
		Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	Units per sec^3 (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented		
	% of Time (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration.		
	Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration.		
	Units per MasterUnits^3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
	% of Time-Master Driven (Master Driven Mode Units)					
	Master Units (Master Driven Mode Units)					

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed is in Master Units.

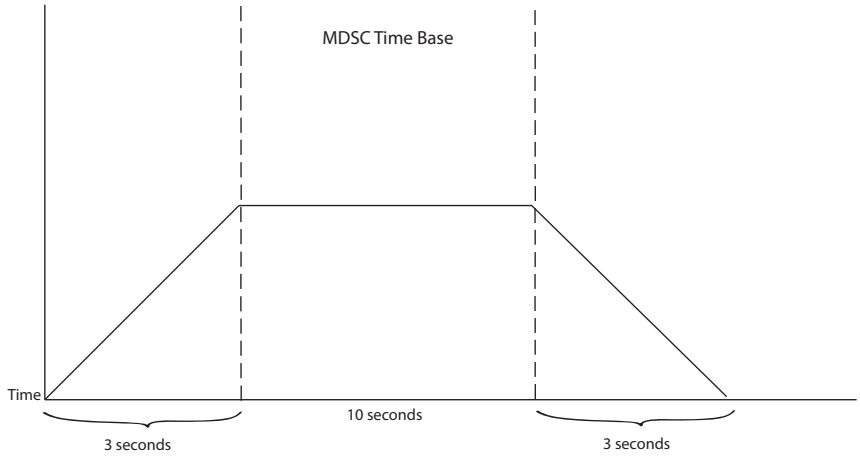
Table 148 - Acceleration Units and Jerk Units when Speed is in Master Units

Speed Units in Master Units		Acceleration (Speed in MasterUnits)				
		Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	Units per sec^3 (Time Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)					
	% of Time (Time Driven Mode Units)					
	Seconds (Time Driven Mode Units)					
	Units per MasterUnits^3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Not Implemented	Not Implemented
	% of Time-Master Driven (Master Driven Mode Units)				Not Implemented	New Enumeration.
	Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration.

Time Based Planning

With time based planning, the dynamics of the move, that is, the Speed, Acceleration, Deceleration, or Jerk can be directly programmed in units of seconds.

Figure 41 - MDSC Time Base Example



If the sum of the Acceleration and Deceleration time is greater than the total time as specified by the speed, the Acceleration and Deceleration time is reduced proportionately so the total specified time in the Speed Parameter is met.

The Speed, when programmed in seconds, takes priority over Acceleration and Deceleration, which in turn takes priority over Jerk. The move will always complete in the specified total time. If the time is too short for the axes physical limits, then a servo drive fault results.

If programmed, move parameters are inconsistent, then an attempt to fix the inconsistency is done by giving priority to parameters in the following order (the lower number has the higher priority):

1. Total move length

   This parameter is always satisfied as programmed, it is never changed.
2. Total move duration
3. Acceleration and Deceleration time
4. Acceleration and deceleration Jerk
   - If the Acceleration Jerk time is greater than 50% of the Acceleration Time, then the time is reduced so that the Acceleration completes in the specified Acceleration Time. Similar calculations are performed for the Deceleration Jerk times.
   - If speed units are programmed in units of seconds, then acceleration and deceleration must also be programmed in seconds. For this case, the jerk can be programmed in either units of seconds or percentage of time.

TIP Time based planning cannot be used for jogged moves (MAJ).

Time Based Planning is only functional for moves starting and ending at zero velocity. An error generates if a move is started with a non-zero velocity or acceleration. You have the option to wake-up a paused or dwelling move with a time based move. See Dwells on page 324 for more information.

The values of zero for acceleration, deceleration, or jerk times are permitted and will generate infinite acceleration, deceleration, or jerk. A value of zero for speed generates a runtime error. For an S-curve profile, an infinite jerk will change to a trapezoidal profile.

All the existing functionality for the Time Based programming mode is supported when you operate in Master Driven mode. Time becomes master distance in Master Driven mode.

IMPORTANT	Time based planning is not implemented for coordinated moves in Logix Designer application, version 20. Dynamics in seconds are incompatible with Merge Speed = Current. This will result in error 94, MDSC_UNITS_CONFLICT.

Dwells

You have the option to program a dwell by using Time Based Programming in either Time Driven mode or Master Driven mode. When a zero length move is programmed, the duration of the dwell is programmed in the Speed parameter and the Acceleration, Deceleration, and Jerk parameters are ignored.

When in time driven mode, the duration of the dwell is programmed in seconds. When in MDSC mode, the duration of the dwell is programmed in units of Master Distance. If speed is specified in Master Units, the move remains active until the specified Master distance has been traversed.

Similarly, when in Time Driven mode, program the move time directly in seconds and with a zero departure. This results in a programmed delay of the specified time.

A zero length move that is programmed with a Speed of 0 Seconds or zero Units per Master Units will complete in the minimum time possible, which is 1 coarse update period.

In MDSC mode, the dwell starts either at the Master Lock Position or immediately, depending on the programmed Lock Direction parameter, and continues for a duration as specified in the Speed parameter.

Time Based Programming Errors

There are two time based programming errors:

- AXIS_NOT_AT_REST, Error 100
- MDSC_UNITS_CONFLICT, Error 94

An AXIS_NOT_AT_REST, (Error 100) occurs if a move is programmed by using Time Based Planning and is started when the active move is at a nonzero velocity. This means that a move by using Time Based Planning with the Merge

Enabled in an instruction will cause an error for most cases because merge is typically used when the axes are moving.

MDSC_UNITS_CONFLICT, (Error 94) occurs if speed is programmed in units of seconds and acceleration, deceleration, or jerk is not programmed in seconds (or % of Time for jerk).

Path Fidelity

The path fidelity (remaining on path) is maintained through the full range of Master dynamics of Speed, Acceleration, Deceleration, and Jerk in Master Driven mode. This means that the move path does not change as the Master speed is changed with an MCD.

There is no end position overshoot as the Master velocity is changed; however, if the Master velocity is changed extremely rapidly, there can be a Slave velocity or acceleration or deceleration overshoot.

Status Bits for Motion Instructions (MAM, MATC, MAJ) When MDAC is Active

The following table describes the predefined data type status bits for motion instruction MAM, MATC, and MAJ.

Table 149 - Status Bits for Motion Instructions when an MDAC Instruction is Active

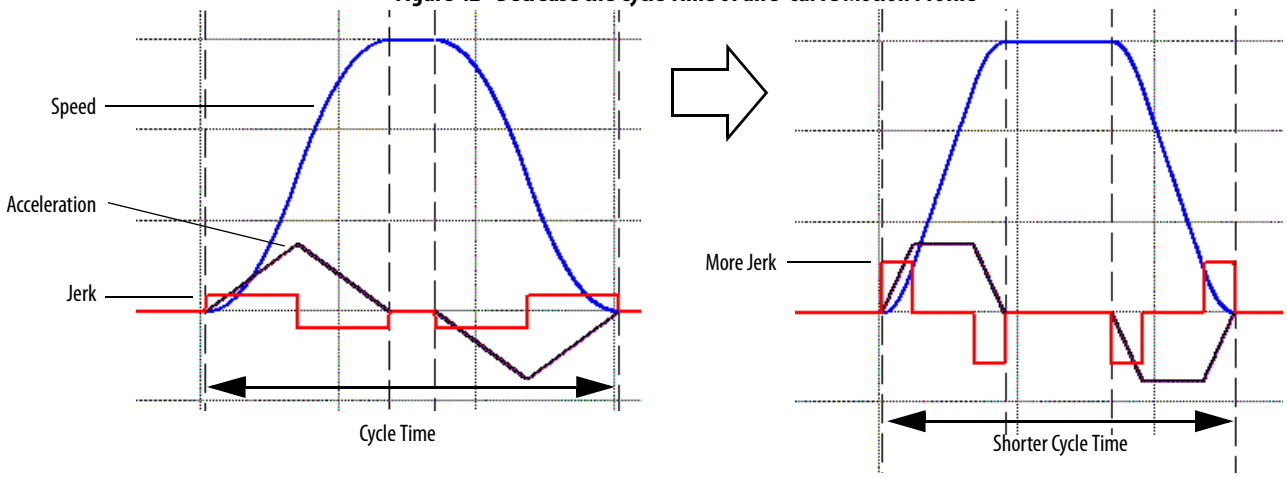
Bit	Description
.EN (Enable)	The Enable bit is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done)	Timing done output. Indicates when the accumulated time is greater than or equal to the preset value.
.ER (Error)	The Error bit is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.PC (Process Complete)	The Process Complete bit is set after the diagnostic test process has been successfully completed.
.IP (In Process)	The In Process bit is set on positive rung transition and cleared after the diagnostic test process is complete, or terminated by a stop command, shutdown, or a servo fault.
.AC (Active)	The Active bit is set when a move (MAJ, MAM or MATC) goes IP in Master Driven mode on the axis that is selected as the Slave Axis of the MDAC instruction. The AC bit will be reset when all single axis motion that is being controlled by the MDAC is completed. If the Slave Axis is started in Time Driven mode, then the AC bit of the MDAC does not go active. The IP bit of the MDAC instruction does not change at this time.
ACCEL	The ACCEL bit is set as expected during motion. It is independent of Master Axis acceleration. The ACCEL bit on the instruction driving the Slave Axis, for example, MAM on the Master Axis is set as the Slave Axis accelerating to its commanded speed as a result of the master axis acceleration. In addition, The ACCEL bit of the slave axis is set when it's accelerating due to not being at it's programmed acceleration. This bit is insensitive to acceleration occurring on the Master Axis. However, the AccelStatus bit, which is in the MotionStatus word of the Slave Axis (not the instruction driving the slave axis), is set or cleared based on changes in the programmed velocity of the Slave Axis.
DECEL	The DECEL bit is set as expected during motion. It is independent of Master Axis deceleration. The DECEL bit on the instruction driving the Slave Axis is set as the Slave Axis is decelerating to its commanded speed. as a result of the master axis deceleration. In addition, The DECEL bit of the slave axis is set when it is decelerating due to not being at its programmed deceleration. This bit is insensitive to deceleration occurring on the Master Axis. However, the DecelStatus bit, which is in the MotionStatus word of the Slave Axis (not the instruction driving the slave axis), is set or cleared based on changes in the programmed velocity of the Slave Axis.
TrackingMaster	Indicates that the Slave Axis is tracking the Master Axis. Only used in Master Driven mode. When an instruction is initiated in Master Driven mode, the Slave Axis accelerates to the speed that its programmed for MDSC mode. This bit is insensitive to slave acceleration/deceleration of the master axis. The Tracking Master is set when the acceleration is complete in MDSC mode. This means that the Slave Axis is synchronized to the Master Axis. This bit is insensitive to the accel/decel of the master axis. The Tracking Master bit is cleared when any of the following occurs on the Slave Axis: • When the Slave Axis starts to either accelerate or decelerate for any reason, for example, for an MCD or an MAS being issued. • When the Slave Axis is relinked to another Master Axis. In this situation, the TrackingMaster bit is first cleared and then it is set again in the new instruction status word when the Slave Axis starts tracking the new Master Axis again. • The Slave Axis is stopped. The Tracking Master is cleared as soon as the stop is initiated on the Slave Axis. This bit is never set when LockDir = NONE. The Tracking Master bit on the Slave Axis is not affected by any operation (for example, MAS, MCD) on the Master Axis. The Tracking Master bit is always cleared in Time Driven mode.
CalculatedDataAvailable	Indicates when the output data in the Calculated Data parameter has been updated and is available. The CalculatedDataAvailable bit is not set for any move that Event Distance is not specified, that is, for any move where the Event Distance parameter in the instruction is zero, this is not the value in the parameter array.

Tune an S-curve Profile

Use this procedure to balance the smoothness and cycle time of motion that uses an S-curve profile.

Perform this procedure when you want to decrease the cycle time of an S-curve motion profile but keep some of the profile's smoothness.

Figure 42 - Decrease the Cycle Time of an S-curve Motion Profile



To use this procedure, your application must meet these requirements:

- The controller is at revision 16 or later.
- One of these instructions produce the motion:
  - Motion Axis Move (MAM)
  - Motion Axis Jog (MAJ)
  - Motion Axis Stop (MAS)
- The instruction uses an S-curve profile.

Procedure

IMPORTANT In this procedure, you increase the jerk. This increases the stress on the equipment and load. Make sure you can identify when the equipment or load has reached its jerk limit.

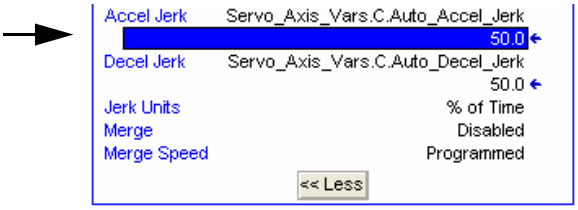
1. Are the Jerk Units set to % of Time?

Table 150 - Increasing the Jerk

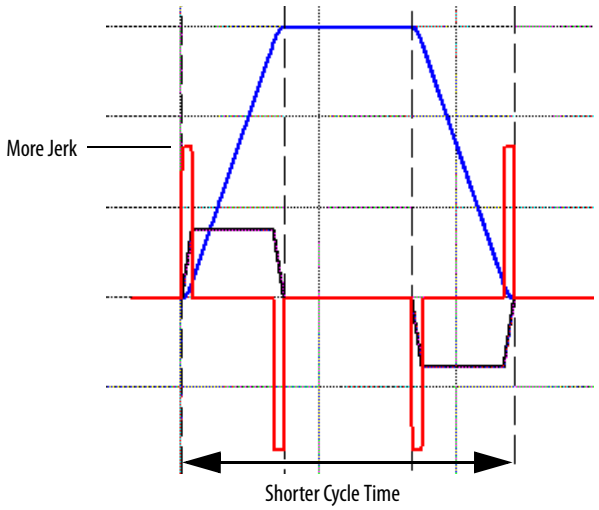
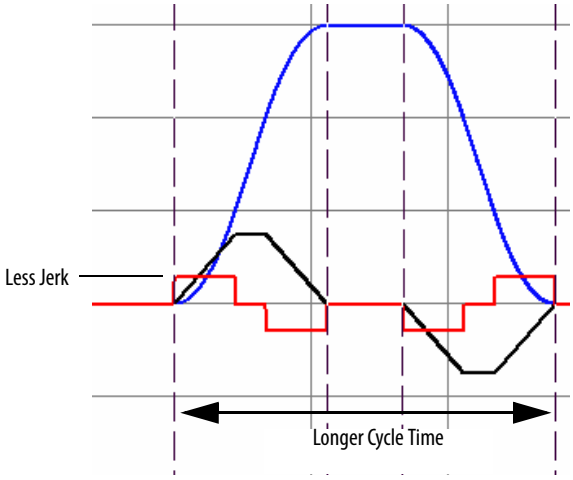
If the Jerk Units are	Then
% of Time 	Continue with step 2.
% of Maximum 	1. Change the Jerk Units to % of Time. 2. Continue with step 2.
Units per sec3 	

2. Set the Jerk values to 50% of Time.

Example



3. Test your equipment and observe its jerk.
4. Adjust the Jerk values.

If there is	Then	Which results in
NOT too much jerk	Reduce the % of Time.	
Too much jerk	Increase the % of Time.	

5. Repeat step 3 and step 4 until you have the desired balance between smoothness and cycle time.

Additional Resources

See the Analyzing Axis Motion section on page 331.

Notes:

Analyzing Axis Motion

Use this information when you are troubleshoot some situations that could happen while you are running an axis.

Topic	Page
Axis Accelerates While Stopping	331
Axis Overshoots the Target Speed	334
Delay If a Jog Is Stopped and Restarted	336
Axis Reverses Direction When Stopped and Started	338
Axis Overshoots Its Position and Reverses Direction	342

Axis Accelerates While Stopping

In some circumstances, while an axis is accelerating, you try to stop it, and the axis keeps accelerating for a short time before it starts to decelerate.

Table 151 - Axis Deceleration

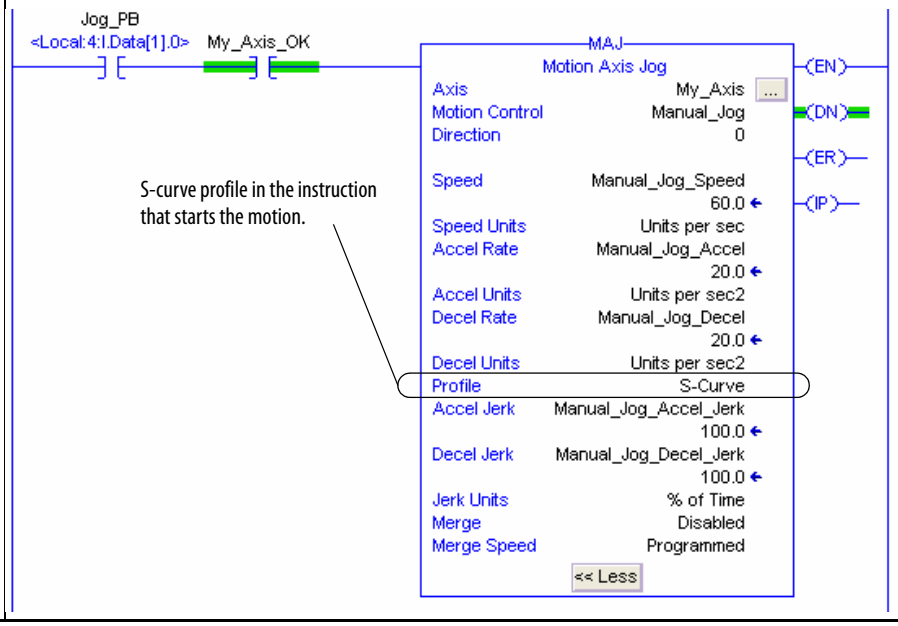
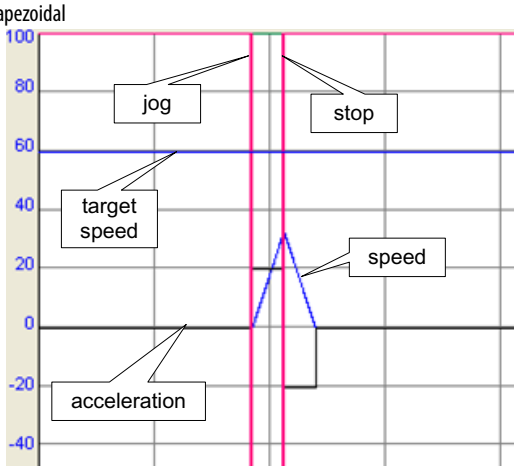
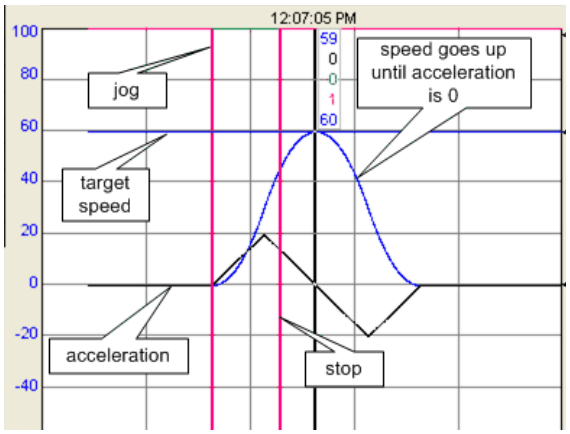
Example	You execute a Motion Axis Jog (MAJ) instruction. Before the axis get s to its target speed, you execute a Motion Axis Stop (MAS) instruction. The axis continues to speed up and then eventually slows to a stop.
Look For	

Table 151 - Axis Deceleration

Cause	When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration. • An S-curve profile has to get acceleration to zero before the axis can slow down. • The time it takes depends on the jerk, acceleration, and speed. • In the meantime, the axis continues to speed up. The following trends show how the axis stops with a trapezoidal profile and an S-curve profile.
	Trapezoidal  The axis slows down as soon as you start the stopping instruction.
	S-curve  The axis continues to speed up until the S-curve profile brings the acceleration to zero.
Corrective Action Revision 15 and earlier	1. Use an Motion Axis Jog (MAJ) instruction to stop the axis. 2. Set the speed of the stopping Motion Axis Jog (MAJ) instruction to zero. 3. Use a higher acceleration in the stopping Motion Axis Jog (MAJ). **Reason:** This increases the acceleration jerk. The axis can begin to stop sooner at the higher acceleration jerk. 4. Use a deceleration that gives you the response you want without too much jerk. **Important:** Use the **same** deceleration in both instructions. Otherwise the axis could reverse directions when you go from stopping to starting again.

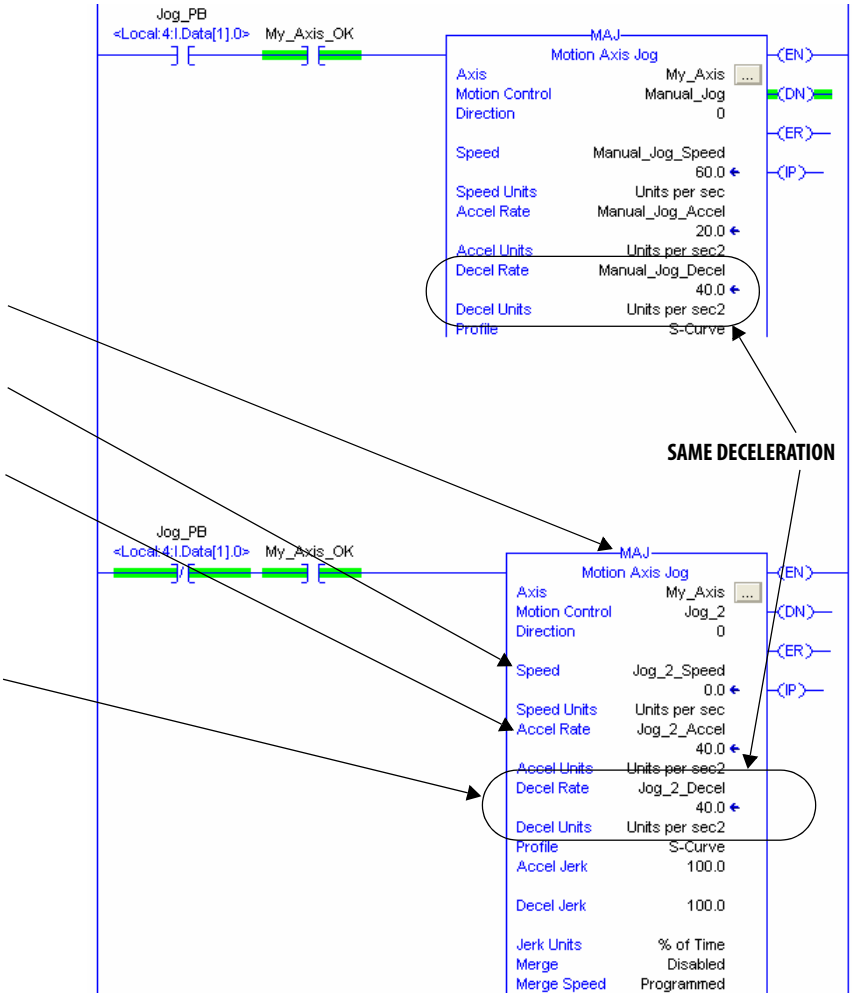


Table 151 - Axis Deceleration

Corrective Action Revision 16 and greater	**Important**: Leave bit 0 of the DynamicsConfigurationBits attribute for the axis turned ON. Otherwise this corrective action won't work. See the Logix Designer online help for more information. Help> Contents> GSV/SSV Objects> Axis> Dynamics Configuration Bits Increase the deceleration jerk of an Motion Axis Stop (MAS) instruction to get a quicker stop. • If the Jerk Units are % of Time, then reduce the % of Time on the Decel Jerk. • If the Jerk Units are % of Maximum, then increase the % of Maximum on the Decel Jerk. • If the Jerk Units are Units per Sec^3, then increase the Units per Sec^3 on the Decel Jerk.

Axis Overshoots the Target Speed

While an axis is accelerating, you try to stop the axis or change its speed. The axis keeps accelerating and goes past its initial target speed. Eventually it starts to decelerate.

IMPORTANT	Revision 16 improved how the controller handles changes to an S-curve profile. See the Logix Designer software online help for more information. Help ⇒ Contents ⇒ GSV/SSV Objects ⇒ Axis ⇒ Dynamics Configuration Bits

Table 152 - Axis Overshoots the Target Speed

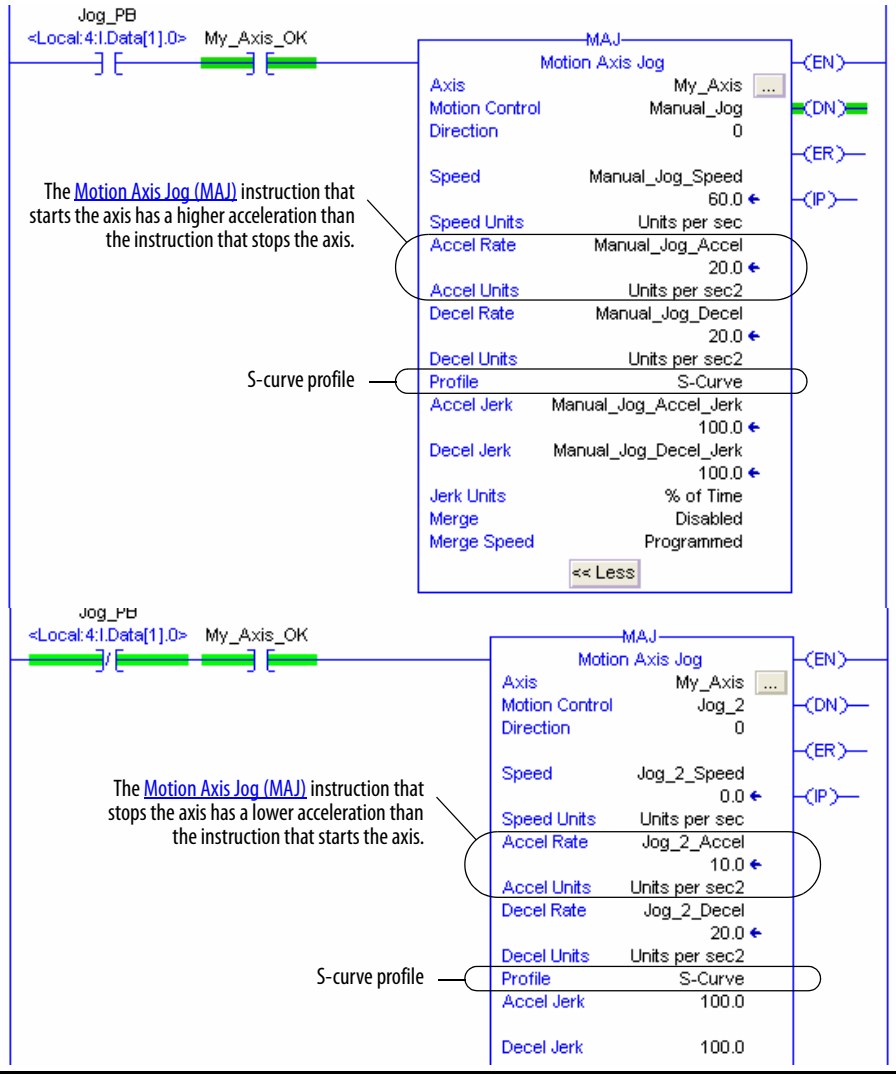
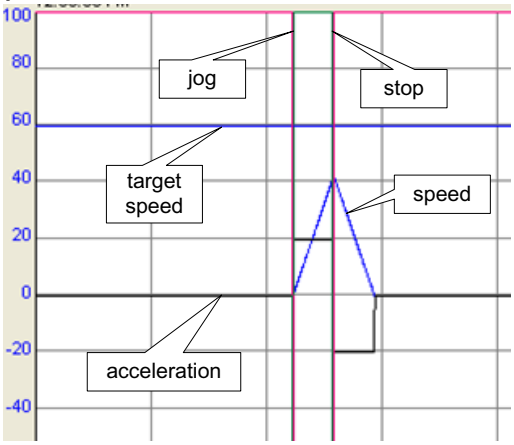
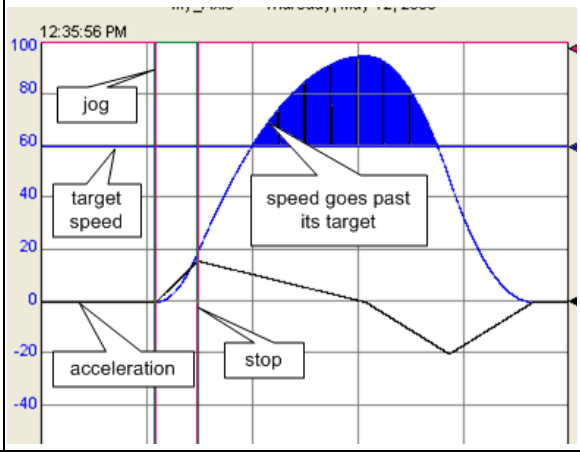
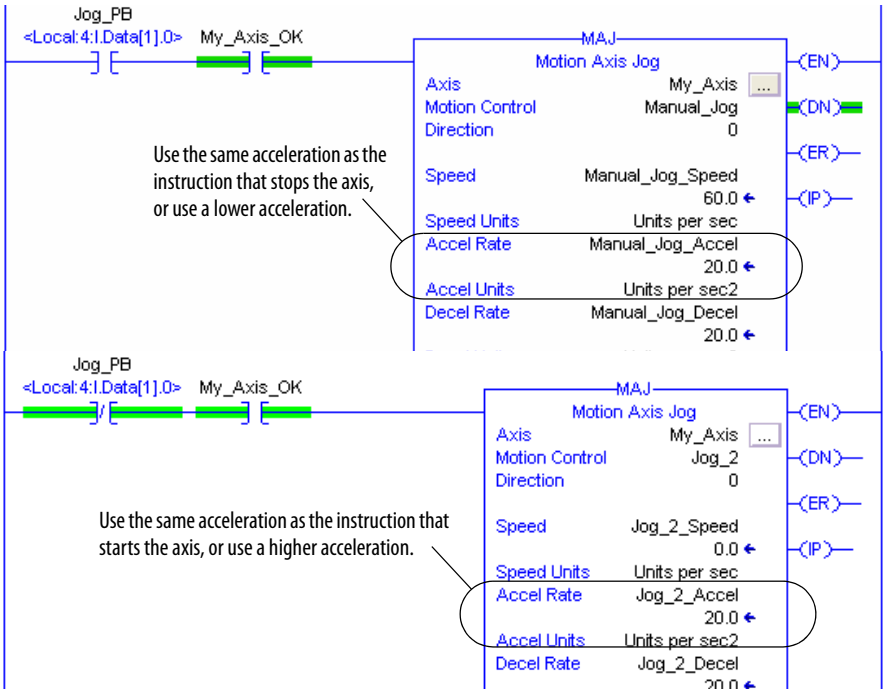
Example	You execute a Motion Axis Jog (MAJ) instruction. Before the axis gets to its target speed, you try to stop it with another Motion Axis Jog (MAJ) instruction. The speed of the second instruction is set to zero. The axis continues to speed up and overshoots its initial target speed. Eventually it slows to a stop.
Look For	

Table 152 - Axis Overshoots the Target Speed

<table>
<tr><td>Cause</td><td colspan="2">When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration.
• When the stopping instruction starts, the controller recalculates jerk and builds a new S-curve profile.
• If the stopping instruction uses a lower acceleration, the jerk is lower. It takes longer at the lower jerk to get acceleration to zero.
• In the meantime, the axis continues past its initial target speed.
The following trends show how the axis stops with a trapezoidal profile and an S-curve profile.</td></tr>
<tr><td></td><td>Trapezoidal
 
</td><td>S-curve
 
</td></tr>
<tr><td></td><td>The axis slows down as soon as you start the stopping instruction. The lower acceleration doesn't change the response of the axis.</td><td>The stopping instruction reduces the acceleration of the axis. It now takes longer to bring the acceleration to zero. The axis continues past its target speed until acceleration equals zero.</td></tr>
<tr><td>Corrective Action</td><td colspan="2">Use a Motion Axis Stop (MAS) instruction to stop the axis.
Or set up your instructions like this.
 
</td></tr>
</table>

Delay If a Jog Is Stopped and Restarted

While an axis is jogging at its target speed, you stop the axis. Before the axis stops completely, you restart the jog. The axis continues to slow down before it speeds up.

Table 153 - Delay If a Jog Is Stopped and Restarted

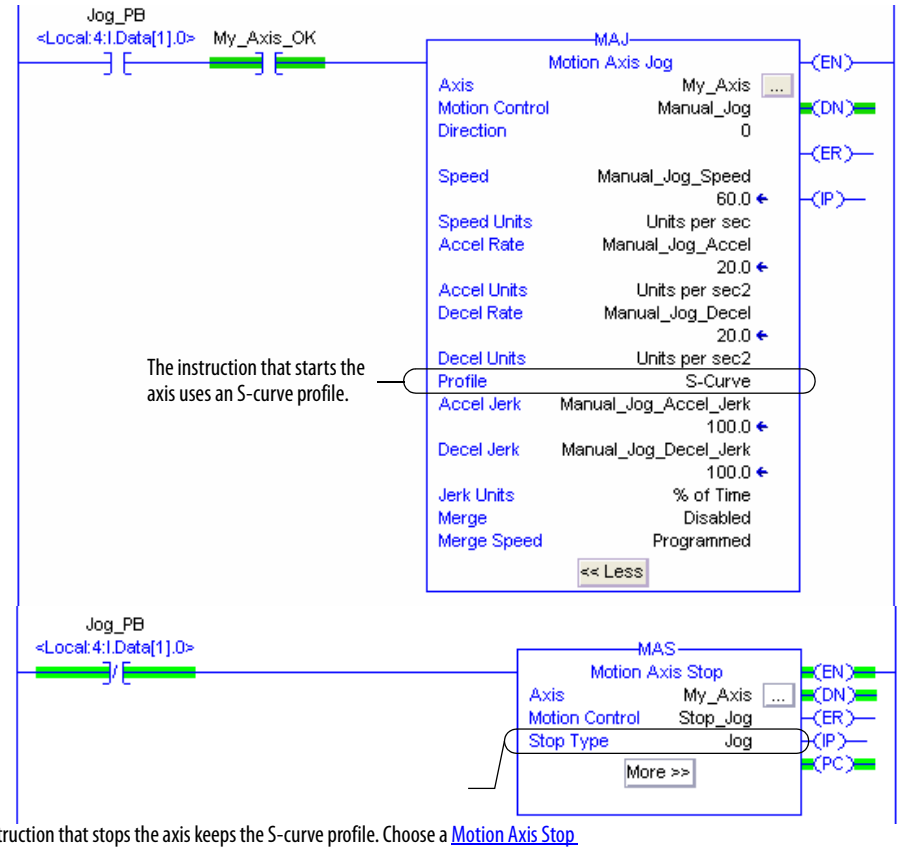
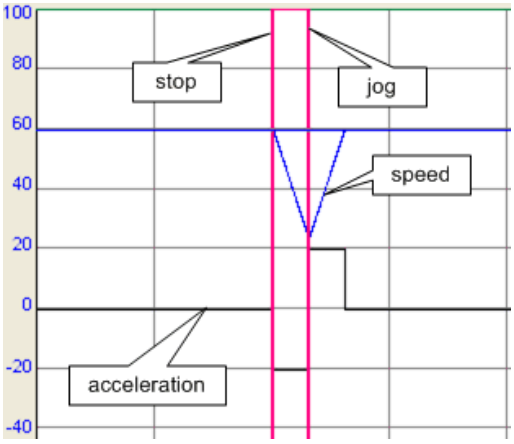
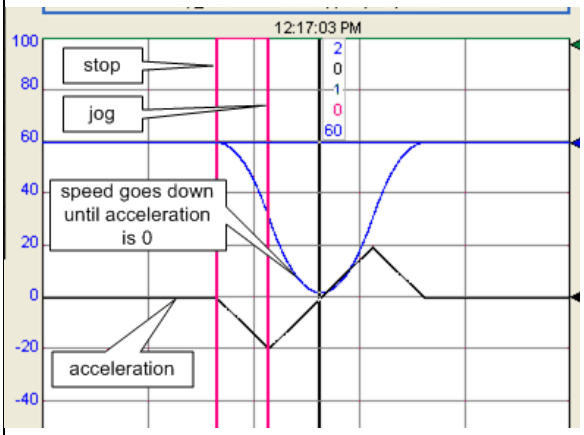
Example	You execute a Motion Axis Stop (MAS) instruction to stop a jog. While the axis is slowing down, you execute a Motion Axis Jog (MAJ) instruction to start the axis again. The axis doesn't respond right away. It continues to slow down. Eventually it speeds back up to the target speed.
Look For	 The instruction that stops the axis keeps the S-curve profile. Choose a Motion Axis Stop (MAS) instruction with the *Stop Type* set to *Jog*. In that case, the axis keeps the profile of the Motion Axis Jog (MAJ) instruction that started the axis.

Table 153 - Delay If a Jog Is Stopped and Restarted

Cause	When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration. • An S-curve profile has to get acceleration to zero before the axis can speed up again. • The axis continues to slow down until the S-curve profile brings the acceleration to zero. The following trends show how the axis stops and starts with a trapezoidal profile and an S-curve profile.	
	Trapezoidal 	**S-curve** 
	The axis speeds back up as soon as you start the jog again.	The axis continues to slow down until the S-curve profile brings the acceleration rate to zero.
Corrective Action	• If your controller is revision 15 or earlier, increase the deceleration rate of the Motion Axis Jog (MAJ) instruction that starts the jog. This increases the deceleration jerk. The axis stops the deceleration sooner at the higher deceleration jerk. • If your controller is revision 16 or later, increase the deceleration jerk of the Motion Axis Jog (MAJ) instruction that starts the jog. The axis stops the deceleration sooner at the higher deceleration jerk.	

Axis Reverses Direction When Stopped and Started

While an axis is jogging at its target speed, you stop the axis. Before the axis stops completely, you restart the jog. The axis continues to slow down and then reverse direction. Eventually the axis changes direction again and moves in the programmed direction.

IMPORTANT	You shouldn't see this situation in revision 16 and later. See Corrective Action Revision 16 and later on page 341.

Table 154 - Axis Reverse Direction

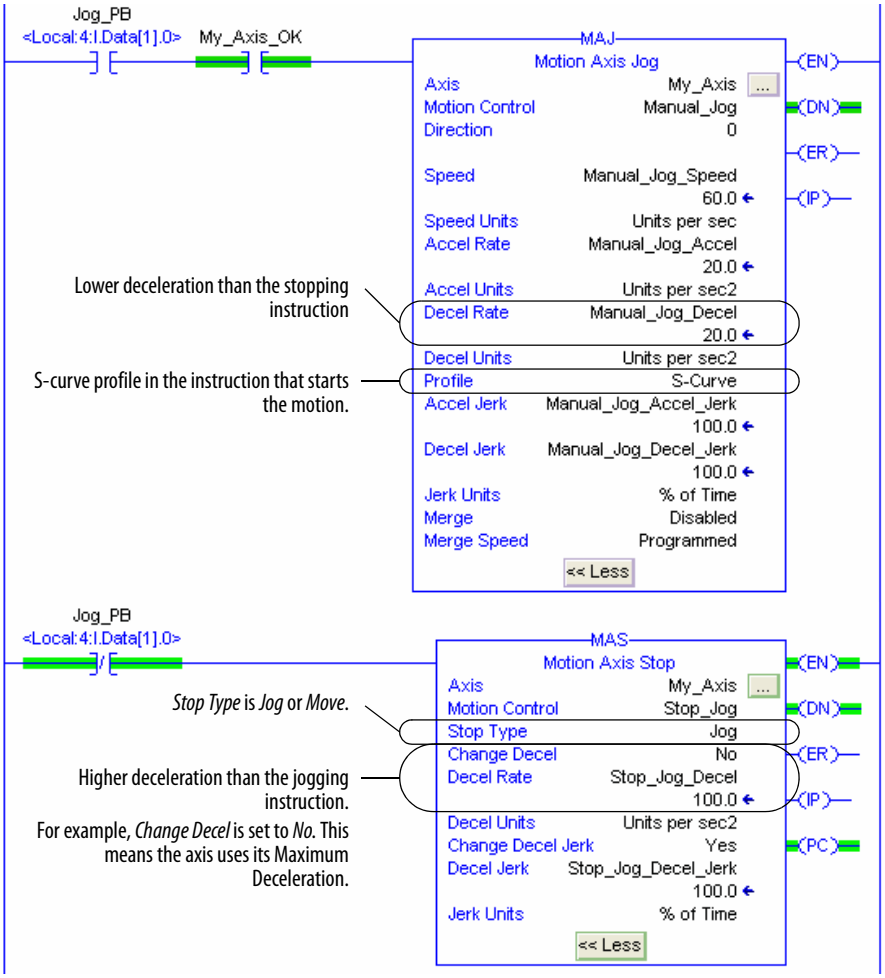
Example	You execute a Motion Axis Stop (MAS) instruction to stop a jog. While the axis is slowing down, you execute a Motion Axis Jog (MAJ) instruction to start the axis again. The axis continues to slow down and then moves in the opposite direction. Eventually goes back to its programmed direction.
Look For	

Table 154 - Axis Reverse Direction

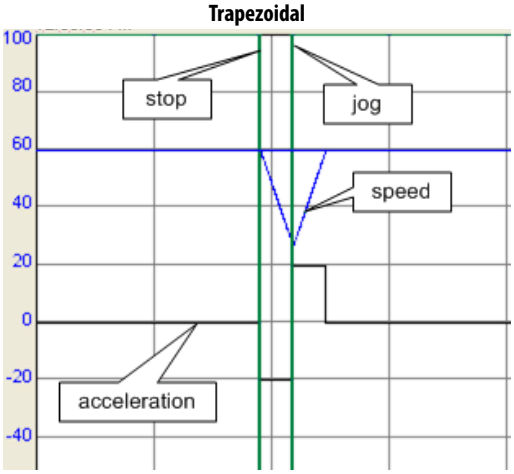
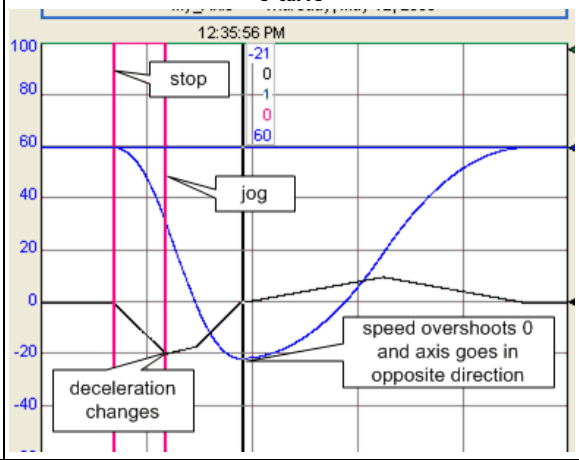
<table>
<tr><td>Cause</td><td colspan="2">When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration.
• When the Motion Axis Jog (MAJ) instruction starts again, the controller recalculates jerk and builds a new S-curve profile.
• If the Motion Axis Jog (MAJ) instruction uses a lower deceleration, the jerk is lower. It takes longer at the lower jerk to get deceleration to zero.
• In the meantime, the axis continues past zero speed and moves in the opposite direction.
The following trends show how the axis stops and starts with a trapezoidal profile and an S-curve profile.</td></tr>
<tr><td></td><td>Trapezoidal
 
</td><td>S-curve
 
</td></tr>
<tr><td></td><td>The axis speeds back up as soon as you start the jog again. The lower deceleration doesn't change the response of the axis.</td><td>The jog instruction reduces the deceleration of the axis. It now takes longer to bring the deceleration to zero. The speed overshoots zero and the axis moves in the opposite direction.</td></tr>
</table>

Table 154 - Axis Reverse Direction

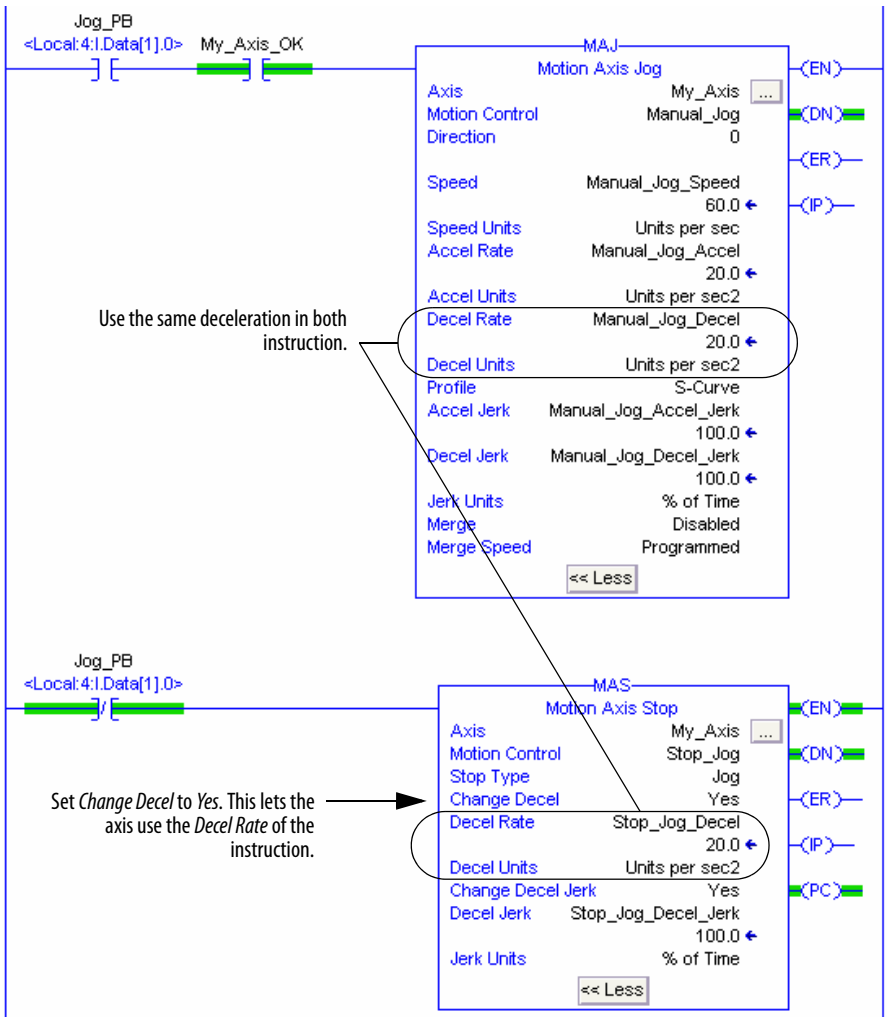
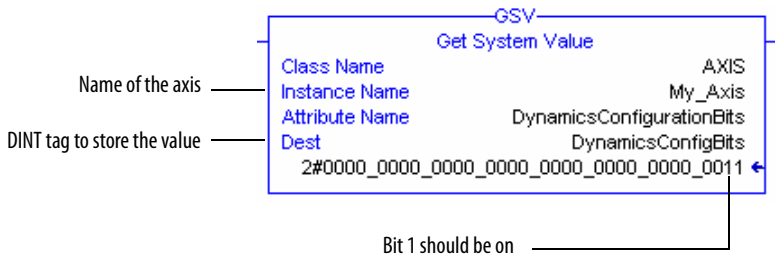
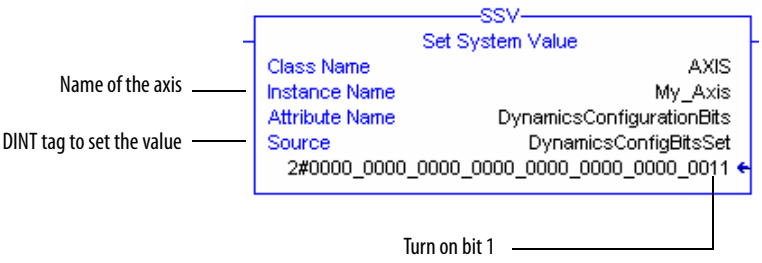
Corrective Action Revision 15 and earlier	

Table 154 - Axis Reverse Direction

Corrective Action Revision 16 and later	If you're still seeing an axis reversal, make sure bit 1 of the DynamicsConfigurationBits for the axis is on. 1. Use a Get System Value (GSV) instruction to see if the algorithm is on.  2. If bit 1 is off, turn it on.  See the Logix Designer online help for more information: Help > Contents > GSV/SSV Objects > Axis > Dynamics Configuration Bits.

Axis Overshoots Its Position and Reverses Direction

While an axis is moving to a target position, you change a parameter of the move. The axis overshoots its target position. Eventually the axis stops and moves back to its target position.

Table 155 - Axis Overshoots

Example	You execute a Motion Change Dynamics (MCD) instruction to reduce the deceleration while a Motion Axis Move (MAM) instruction is in process. The axis continues past the target position of the move, stops, and returns to the target position.
Look For	Motion Change Dynamics (MCD) instruction that reduces the deceleration of the move

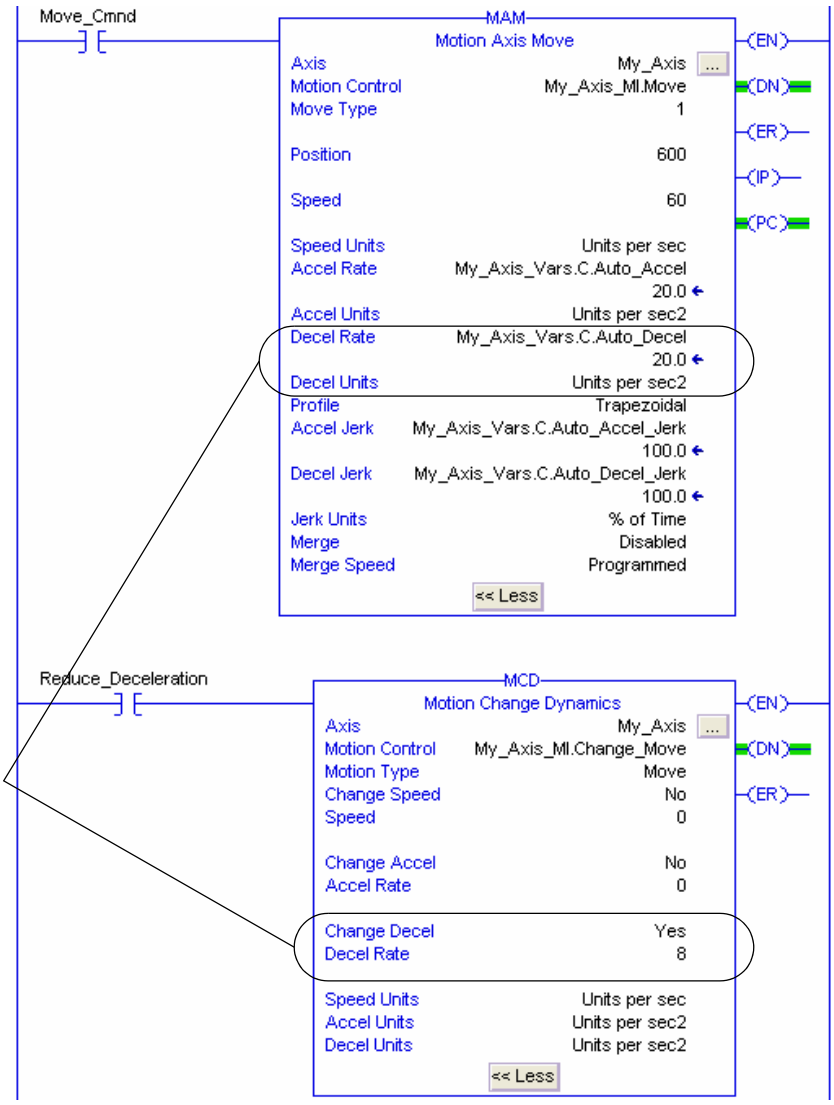
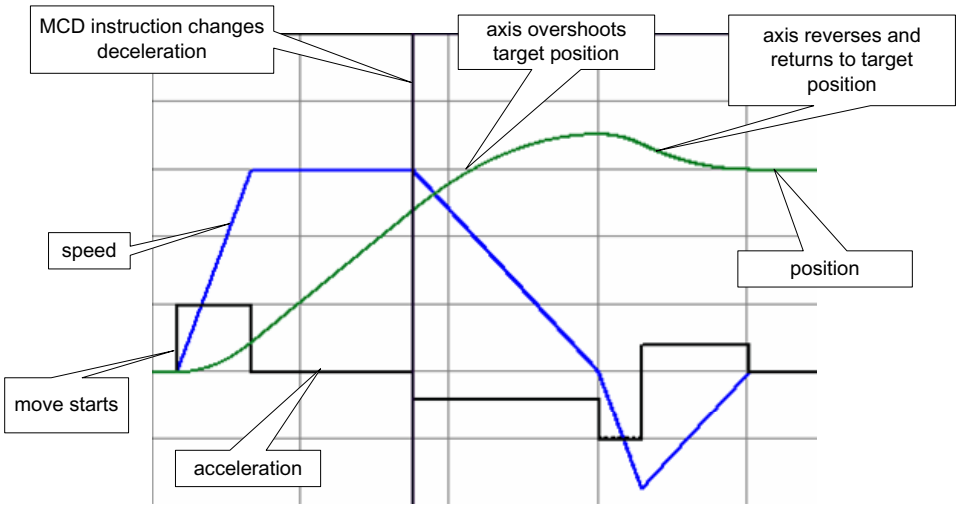


Table 155 - Axis Overshoots

Cause	The axis doesn't have enough time at the new lower deceleration to stop at the target position. It stops past the target position. Then it corrects to get back to the target position. 
Corrective Action	To avoid overshooting position, either: • avoid decreasing the deceleration or deceleration jerk while an axis is decelerating along an S-curve profile. • avoid increasing the programmed speed while an axis is decelerating along an S-curve profile. This has the same effect as decreasing the deceleration jerk. • test any changes in small increments to make sure a change doesn't cause an overshoot during normal operation.

Notes:

Error Codes (ERR) for Motion Instructions

This table lists the error codes for Logix Designer software motion instructions.

Table 156 - Motion Instruction Error Codes Descriptions

Error	Corrective Action or Cause	Notes
1	Reserved Error Code 1	Reserved for future use.
2	Reserved Error Code 2	Reserved for future use.
3	Look for another instance of this type of instruction. See if its EN bit is on but its DN and ER bits are off (enabled but not done or erred). Wait until its DN or ER bit turns on.	Execution Collision You cannot execute an instruction by using the same control word as another instruction if the other instruction is not done or has an error. Regardless, it is recommended that each instruction has a unique control word.
4	Open the servo loop before you execute this instruction.	Servo On State Error
5	Close the servo loop before you execute this instruction.	Servo Off State Error For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
6	Disable the axis drive.	Drive On State Error
7	Execute a Motion Axis Shutdown Reset (MASR) instruction or direct command to reset the axis.	Shutdown State Error For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
8	The configured axis type is not correct.	Wrong Axis Type For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
9	The instruction tried to execute in a direction that aggravates the current overtravel condition.	Overtravel Condition
10	The master axis reference is the same as the slave axis reference or the Master Axis is also an axis in the Save Coordinate System.	Master Axis Conflict
11	At least one axis is not configured to a physical motion module or has not been assigned to a Motion Group.	Axis Not Configured For single axis instructions: the Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis For the MAM, MCD, and MAJ instructions in time driven mode, the axis being moved is a slave axis. For multi-axes instructions: the Extended Error code for MDCC, MCLM, MCCM, and MCCD is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis
12	Messaging to the servo module failed.	Servo Message Failure

Table 156 - Motion Instruction Error Codes Descriptions (Continued)

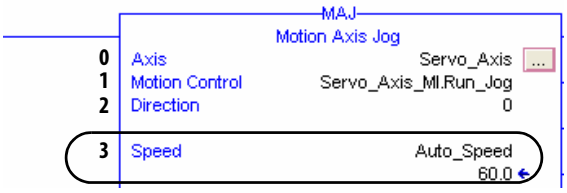
Error	Corrective Action or Cause	Notes
13	Look at the extended error code (EXERR) for the instruction. It identifies an operand that is outside its range. **Example**: An MAJ instruction has an ERR = 13 and an EXERR = 3. In this case, change the speed so that it's in range. 	Parameter Out Of Range An EXERR = 0 means the first operand of the instruction is outside its range.
14	The instruction cannot apply the tuning parameters because of an error in the run tuning instruction.	Tune Process Error
15	The instruction cannot apply the diagnostic parameters because of an error in the run diagnostic test instruction.	Test Process Error
16	Wait until the homing process is done.	Home In Process Error
17	The instruction tried to execute a rotary move on an axis that is not configured for rotary operation.	Axis Mode Not Rotary
18	The axis type is configured as unused.	Axis Type Unused
19	The motion group is not in the synchronized state. This could be caused by a missing or mis-configured servo module.	Group Not Synchronized Group sync status is only cleared on a group overlap or CST loss fault.
20	The axis is in the faulted state.	Axis In Faulted State
21	The group is in the faulted state.	Group In Faulted State
22	Stop the axis before you execute this instruction.	Axis In Motion
23	An instruction attempted an illegal change of dynamics.	Illegal Dynamic Change
24	Take the controller out of test mode.	Illegal Controller Mode
25	You attempted to execute an instruction that is not correct.	Illegal Instruction
26	The cam array is of an illegal length.	Illegal Cam Length
27	The cam profile array is of an illegal length.	Illegal Cam Profile Length
28	You have an illegal segment type in the cam element.	Illegal Cam Type
29	You have an illegal order of cam elements.	Illegal Cam Order
30	You tried to execute a cam profile while it is being calculated.	Cam Profile Being Calculated
31	The cam profile array you tried to execute is in use.	Cam Profile Being Used
32	The cam profile array you tried to execute has not been calculated.	Cam Profile Not Calculated
33	A MAM - Master Offset move was attempted without a Position CAM in process.	Position Cam Not Enabled
34	A MAH instruction is trying to start while a registration is already running.	Registration in Progress
35	The specified execution target exceeds the number of Output Cam targets configured for the axis.	Illegal Execution Target

Table 156 - Motion Instruction Error Codes Descriptions (Continued)

Error	Corrective Action or Cause	Notes
36	Either the size of the Output Cam array is not supported or the value of one of its members is out of range.	Illegal Output Cam • ExErr#1: Output bit less than 0 or greater than 31. • ExErr#2: Latch type less than 0 or greater than 3. • ExErr#3: Unlatch type less than 0 or greater than 5. • ExErr#4: Left or right position is out of cam range and the latch or unlatch type is set to `Position' or `Position and Enable'. • ExErr#5: Duration less than or equal to 0 and the unlatch type is set to `Duration' or `Duration and Enable'. • ExErr#6: Enable type less than 0 or greater than 3 and the latch or unlatch type is set to `Enable', `Position and Enable', or `Duration and Enable'. • ExErr#7: Enable bit less than 0 or greater than 31 and the latch or unlatch type is set to `Enable', `Position and Enable', or `Duration and Enable'. • ExErr#8: Latch type is set to `Inactive' and unlatch type is set to either `Duration' or `Duration and Enable'.
37	Either the size of the Output Compensation array is not supported or the value of one of its members is out of range. The array index associated with errors 36 and 37 are stored in .SEGMENT of the Motion Instruction data type. Only the first of multiple errors are stored. The specific error detected is stored in Extended Error Code (see `Topic Template - No TOC' on page 36). With the ability to dynamically modify the Output Cam table, the Illegal Output Cam error 36 can occur while the MAOC is in-process. In general, the cam elements where an error was detected will be skipped. The following are exceptions and will continue to be processed. • Error 2, Latch Type Invalid. Latch Type defaults to Inactive. • Error 3, Unlatch Type Invalid. Unlatch Type defaults to Inactive. • Error 8, with Unlatch Type of Duration and Enable. Will behave as an Enable Unlatch type.	Illegal Output Compensation • ExErr#1: Mode less than 0 or greater than 3. • ExErr#2: Cycle time less than or equal to 0 and the mode is set to `Pulsed' or `Inverted and Pulsed'. • ExErr#3: Duty cycle less than 0 or greater than 100 and the mode is set to `Pulsed' or `Inverted and Pulsed'.
38	The axis data type is illegal. It is incorrect for the operation.	Illegal Axis Data Type For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
39	You have a conflict in your process. Test and Tune cannot be run at the same time.	Process Conflict
40	You are trying to run a MSO or MAH instruction when the drive is locally disabled.	Drive Locally Disabled
41	The Homing configuration is illegal. You have an absolute homing instruction when the Homing sequence is not immediate.	Illegal Homing Configuration
42	The MASD or MGSD instruction has timed out because it did not receive the shutdown status bit. Usually a programmatic problem caused when either MASD or MGSD is followed by a reset instruction that is initiated before the shutdown bit has been received by the shutdown instruction.	Shutdown Status Timeout
43	You have tried to activate more motion instructions than the instruction queue can hold.	Coordinate System Queue Full
44	You have drawn a line with three 3 points and no centerpoint viapoint or plane centerpoint can be determined.	Circular Collinearity Error
45	You have specified one 1 point radius or "drawn a line" centerpoint, viapoint and no centerpoint radius or plane centerpoint, viapoint can be determined.	Circular Start End Error
46	The programmed centerpoint is not equidistant from start and end point.	Circular R1 R2 Mismatch Error
47	Contact Rockwell Automation Support.	Circular Infinite Solution Error
48	Contact Rockwell Automation Support.	Circular No Solutions Error
49	$\|R\| < 0.01$. R is basically too small to be used in computations.	Circular Small R Error
50	The coordinate system tag is not associated with a motion group.	Coordinate System Not in Group
51	You have set your Termination Type to Actual Position with a value of 0. This value is not supported.	Invalid Actual Tolerance
52	At least one axis is currently undergoing coordinated motion in another coordinate system.	Coordination Motion In Process Error

Table 156 - Motion Instruction Error Codes Descriptions (Continued)

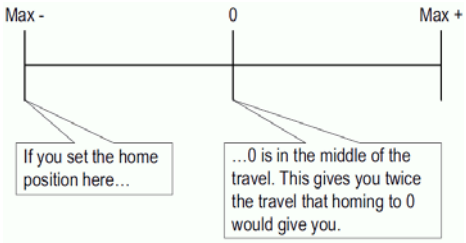
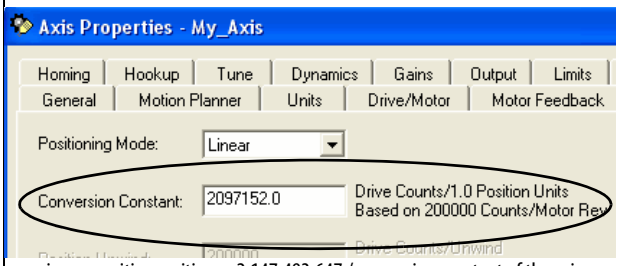
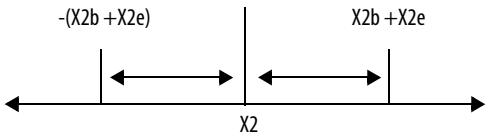
Error	Corrective Action or Cause	Notes
53	Uninhibit the axis.	Axis Is Inhibited For single axis instructions, the Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis For the MAM, MCD, and MAJ instructions in time driven mode, the axis being moved is a slave axis. For multi-axes instructions, the Extended Error code for MDCC, MCLM, MCCM, and MCCD is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis
54	1. Open the properties for the axis. 2. On the Dynamics tab, enter a value for the Maximum Deceleration.	Zero Max Decel You cannot start motion if the maximum deceleration for the axis is zero.
61	See the extended error code (EXERR) for the instruction.	Connection Conflict
62	Cancel the transform that controls this axis or don't use this instruction while the transform is active.	Transform In Progress You cannot execute this instruction if the axis is part of a active transform.
63	Cancel the transform that controls this axis or wait until the transform is done moving the axis.	Axis In Transform Motion You cannot execute this instruction if a transform is moving the axis.
64	Use a Cartesian coordinate system.	Ancillary Not Supported You cannot use a non-Cartesian coordinate system with this instruction.
65	The axis moved too far and the controller cannot store the position. To prevent this error, Set up soft travel limits that keep the axis within the position range. One way to get more travel is to use the max negative or max positive position as your home position. **Example**  **Important**: This error code does not apply to a CIP axis. You will get this error when an absolute MAM instruction is executed when the axis is beyond the max travel limit.	Axis Position Overflow The range for position depends on the conversion constant of the axis.  maximum positive position = 2,147,483,647 / conversion constant of the axis. maximum negative position = -2,147,483,648 / conversion constant of the axis. Choose a conversion constant of 2,097,152 counts/inch. In this case: maximum positive position = 2,147,483,647 / 2,097,152 counts/inch = 1023 inches. maximum negative position = -2,147,483,648 / 2,097,152 counts/inch = -1023 inches. For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. ExErr#1: Axis 0 Caused the Error ExErr#2: Axis 1 Caused the Error ExErr#1: Axis 2 Caused the Error
66	Be sure to keep the robot in the arm solution that you configured it in. You can configure the robot in either a left arm or right arm solution.	You are attempting to fold back an articulated independent or dependent two axis robot on itself at the quadrant boundaries.
67	Change the target positions to values that are within the reach of the robot. If X2b +X2e isn't zero, stay out of this region.  	Invalid Transform Position You're trying to move to a place the robot cannot reach. MCT attempted while at origin. To avoid having the robot fold back on itself or extend beyond its reach, joint limits are calculated internally by the firmware for Delta2D, Delta3D and SCARA Delta robots. If you try and configure a move that violates these limits, this error occurs.

Table 156 - Motion Instruction Error Codes Descriptions (Continued)

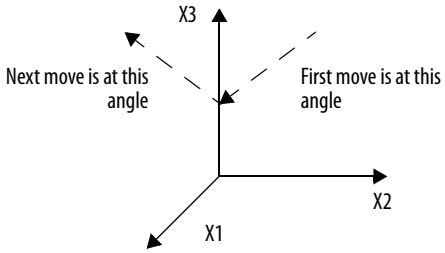
Error	Corrective Action or Cause	Notes
68	Move the joints so that the end of the robot isn't at the origin of the coordinate system.	Transform At Origin You cannot start the transform if the joint angles result in X1 = 0 and X2 = 0.
69	Check the maximum speed configuration of the joints. Use target positions that keep the robot from getting fully stretched or folding back on itself at the origin of the coordinate system. Move in a relatively straight line through positions where X1 = 0 and X2 = 0.	Max Joint Velocity Exceeded The calculated speed is very high. This happens when the robot either: • gets fully stretched. • folds back on itself. • moves away from X1 = 0 and X2 = 0 in a different angle than it approached that position. • is configured with the wrong velocity limit. **Example**: These moves produce this error. 
70	Look for source or target axes that are configured as rotary positioning mode. Change them to linear positioning mode.	Axes In Transform Must Be Linear A transform works only with linear axes.
71	Wait until the transform that you are canceling is completely canceled.	Transform Is Canceling
72	Check the target positions. A calculated joint angle is beyond +/- 360·.	Max Joint Angle Exceeded
73	Check that each MCT instruction in this chain is producing valid positions.	Coordinate System Chaining Error This MCT instruction is part of a chain of MCT instructions. There is a problem with one of the instructions in the chain.
74	Change the orientation to angles that are within +/- 360·.	Invalid Orientation Angle
75	You cannot use this instruction with a SoftLogix controller.	Instruction not supported.
76	1. Open the properties for the axis. 2. On the Dynamics tab, enter a value for the maximum deceleration jerk.	Zero Max Decel Jerk You cannot start motion that uses an S-curve profile if the maximum deceleration jerk for the axis is zero.
77	How many axes are in your coordinate system? 2 — Use a non-mirror transform direction. 3 — Use a non-inverse transform direction.	Transform Direction Not Supported • You're trying to use the mirror directions with a 3-axis coordinate system and a non-zero base offset (X2b) or effector offset (X2e). • Mirror directions are not supported for 2-axis Coordinate Systems. • You are attempting to use either a 2 or 3-axis Cartesian, Delta2D, Delta3D or SCARA Delta target coordinate system with transform directions other than forward and inverse. • You can use inverse mirror directions only when both these conditions are true: • You have a 3-axis coordinate system. • The base offset (X2b) and end effector offset (X2e) of the X2 dimension are zero.
78	New check for a secondary Instruction overlap on top of an active Stop instruction.	Not Allowed While Stopping You cannot overlap certain Motion instructions while stopping. Wait for the first instruction to complete before starting the second instruction. For more information, see the table on page 351 and page 352.
79	Home your axis again. Error of Home instruction occurs, if any other motion on the axis is encountered during the homing sequence.	Internal Homing Sequence Error Invalid Planner State If you see this error, rehome your axis in your application program. Make sure the axis is stopped before home is attempted. If the error persists, contact Rockwell Automation Support.

Table 156 - Motion Instruction Error Codes Descriptions (Continued)

Error	Corrective Action or Cause	Notes
80	When referencing a Scheduled Output Module, for example, the OB16IS, make sure that the Output operand of the MAOC references O:Data, and that the Scheduled Output Module's communication format is set to `Scheduled Output Data per point'. Bad Connection Parameter - Connection Instance Failure. Internal error can occur. Bad Communication Format - I/O subsystem Failure. CIP Sync not synchronized - Scheduled output module reporting not synchronized to a CIP Sync master. Grandmaster Clock mismatch - Scheduled output module has different Grandmaster clock than the controller.	MAOC Invalid Output Operand If the MAOC output operand references an OB16IS Scheduled Output module, two additional checks occur when the MAOC is initiated. The Output operand must be referencing the beginning of the module's output data tag, '0.Data'. The communications format of the OB16IS module must be the default "Scheduled Output Data per Point". If either of these checks fail you see this error. ExErr#1: Invalid Data Tag Reference - The Output operand is not pointing to the 0.Data element of the module's output data tag. ExErr#2: Invalid OB16IS Module Communications Format – The OB16IS communications format has been changed from it's default Scheduled Output Data per Point.
81	Error on MGSR, if a MASD or MGS (programmed) is executed while the MGSR is still in process. Do not overlap the MASD instruction or MGS stop instruction with Stop Mode = Programmed on an active MGSR instruction.	Partial Group Shutdown Reset. If your application program is actively executing an MGSR instruction and you try to execute an MASD instruction or MGS stop instruction with Stop Mode = Programmed on one of the axes affected by the active MGSR instruction, you will see this error on the MGSR instruction.
82	The axis was found to be in the incorrect operational axis state.	CIP axis in incorrect state.
83	The MDS instruction cannot be performed due to control mode selection.	Illegal Control Mode or Method. the MDS instruction is not valid in Position Loop or Feedback Only modes.
84	The CIP drive device digital input is not assigned.	Drive Digital Input Not Assigned
85	Instruction not allowed when redefine position is in process. Performing MAH while MRP is in process results in this instruction error.	Homing, redefine position in progress An Active Redefine Position instruction is in process. You would get this error if any of the motion planner instructions are executed while MRP is in progress. Motion Instructions included are: MAM, MAJ, MCLM, MCCM, MATC, MAPC, MDAC, and MDCC
86	Current use of the MDS instruction requires an optional attribute that is not supported.	Optional attribute not supported by the integrated motion drive being used. Executing a MDS instruction on a CIP- Velocity Loop with Feedback axis associated with a Kinetix 6500 drive errors. The instruction requires an optional attribute that is not currently supported. The MDS instruction is not supported by the drive type.
87	The instruction is invalid while running direct controlled motion.	Not Allowed While In Direct Motion
88	The instruction is invalid while running planned motion.	Not Allowed While Planner Active
93	A move was programmed in MDSC mode before the MDSC link has been established by the execution of a MDAC or MDCC.	MDSC Not Activated
94	• Some dynamics units belong to Master Driven Mode and some to Time Driven Mode. • Some units are time based whereas others are velocity based, for example, Speed in Seconds and Acceleration in units/sec^2. • Incompatibility of units. Dynamics in Seconds are incompatible with Merge Speed = Current.	MDSC Units Conflict
95	• All instructions in the queue must use a compatible Lock Direction, for example, Position Forward Only and Immediate Forward Only. • Lock Direction = None and speed units belong to Master Driven Mode.	MDSC Lock Direction Conflict If you change from Time Driven mode to Master Driven mode while an axis is moving and Lock Direction is not Immediate Forward or Reverse you will get error 95 MDSC Lock Direction Conflict.
96	MDAC (All) and MDAC (something other than All) on the same slave.	MDSC MDAC All Conflict
97	Trying to replace a running Master with a new Master whose speed is zero, or replacing a Slave that is moving via an MAM with another MAM with the same or a different Master that is not moving.	MDSC Idle Master and Slave Moving
98	The actual direction of master axis' motion does not match the direction programmed by Lock Direction parameter (IMMEDIATE FORWARD ONLY or IMMEDIATE REVERSE ONLY) when the slave is already moving.	MDSC Lock Direction Master Direction Mismatch

Table 156 - Motion Instruction Error Codes Descriptions (Continued)

Error	Corrective Action or Cause	Notes
99	Parameter combination not supported.	Feature Not Supported • Performing MDCC on non-Cartesian coordinate system • Using Lock Position for MATC in Time Driven Mode
100	If speed is in seconds or Master units, move must start from rest.	Axis Not At Rest
101	Return data array is either nonexistent or not big enough to store all the requested data.	MDSC Calculated Data Size Error
102	Attempt to activate a second MDSC instruction with a Lock Position or a Merge with a Lock Position while an axis is moving.	MDSC Lock While Moving
103	If the Master Axis is changed and the new slave speed is less than 5% of the original slave speed for Single Axis instructions, or 10%, depending on the move of the original Slave Coordinate System speed, then this error will occur and the change will not be allowed. The same applies when changing from Time Driven mode to MDSC mode.	MDSC Invalid Slave Speed Reduction
104	IF: a motion instruction performs either: • A change in the Master Axis • A change in speed units AND: if in the same update period, the instruction is either forced to pause with a speed of zero, or stopped with a MAS or MCS THEN: the velocity profile is changed to trapezoidal and this error code is reported.	MDSC 2 Instructions were started in 1 Update Period therefore Jerk was Maximized.
105	An instruction in the coordinated motion queue is either trying to change the Master Axis or changing the mode from MDSC mode to Time Driven mode or from Time Driven mode to MDSC mode.	MDSC Invalid Mode Or Master Change
106	Change merge speed parameter.	Can not use Merge to Current when programming in time driven mode using seconds or master driven mode using master units.
107	Target device does not support the requested operation, service, or both.	There is not any corrective action that can be taken.
108	Coordinated System contains a multiplexed axis.	Motion coordinated instructions cannot contain multiplexed axes.

You will get an error if certain Motion Instructions overlap while Motion Stop Instructions are active. In this case, an instruction is actively stopping and a second instruction is initiated that overlaps the active instruction. Table 157 lists some of the overlap instances that will generate errors.

In this case:

- Error # 7 = Shutdown State Error.
- Error #61, ExErr #10 = Connection Conflict, Transform Axes Moving or Locked By Other Operation.
- Error #78 = Not Allowed While Stopping.

The following table lists additional overlap instances that will generate errors.

Table 157 - Generated Errors in Overlap Instances

	Active Stopping Instruction								
	MGS			**MGSD**	**MCS**			**MAS**	
Initiated Second Instruction	Stop Mode = Fast Stop	Stop Mode = Fast Disable	Stop Mode = Programmed		Stop Type = Coordinated Move	Stop Type = Coordinated Transform	Stop Type = All	All Stop Types Except StopType = All	Stop Type = All
MAAT	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MRAT	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAHD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MRHD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAH	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAJ	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MAM	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MAG	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAPC	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MATC	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MDO	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCT	Error #78	Error #78	Error #78	Error # 7	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10
MCCD	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCLM/MCCM (Merge = Disabled)	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78		Error #78
MCLM/MCCM (Merge=Enabled)	Error #78	Error #78	Error #78	Error # 7		Error #78	Error #78		Error #78

Table 158 - Additional Generated Errors in Overlap Instances

		Active Stopping Instruction						
		MGS			**MGSD**	**MCS**	**MAS**	**MASD**
Initiated Second Instruction	Stop Type	Stop Mode = Fast Stop	Stop Mode = Fast Disable	Stop Mode = Programmed	None	Stop Type = All	Stop Type = All	None
MGS	Stop Mode = Fast Stop	Error #78	Error #78	Error #78	Error #7			
	Stop Mode = Fast Disable	Error #78	Error #78	Error #78	Error #7			
	Stop Mode = Programmed	Error #78	Error #78	Error #78	Error #7			
MGSR	None	Error #78	Error #78	Error #78	Error #7			Error #7
MCS	Stop Type = Coordinated Move	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	
	Stop Type = Coordinated Transform	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	
	All Stop Types Except StopType = All	Error #78	Error #78	Error #78	Error #7			
MAS	Stop Type!= All	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	Error #7
	Stop Type = All	Error #78	Error #78	Error #78	Error #7			Error #7
MASR	None	Error #78	Error #78	Error #78	Error #7			Error #7

Motion-related Data Types (Structures)

Use this appendix for information about the following motion-related data types.

Data type	Page
CAM Structure	353
CAM_PROFILE Structure	353
MOTION_GROUP Structure	354
MOTION_INSTRUCTION Data Type	356
OUTPUT_CAM Structure	357
OUTPUT_COMPENSATION Structure	358

CAM Structure

The CAM data type consists of slave and master point pairs as well as an interpolation type. Because there is no association with a specific axis position or time, the point values are unit-less. The interpolation type can be specified for each segment as either linear or cubic. The format of the cam array element is shown in the following table.

Table 159 - CAM Array Element Format Descriptions

Enumerations	Data Type	Description	
MASTER	REAL	The x value of the point.	
SLAVE	REAL	The y value of the point.	
Segment Type	DINT	The type of interpolation.	
		Value	**Description**
		0	linear
		1	cubic

CAM_PROFILE Structure

The CAM_PROFILE data type is an array of coefficients representing a calculated cam profile that can be used as input to a time cam or position cam instruction. The only element available to you is Status that is defined in the following table.

Table 160 - Status Element Description

Enumerations	Data Type	Description	
Status	DINT	The status parameter is used to indicate that the Cam Profile array element has been calculated. If execution of a camming instruction is attempted using an uncalculated element in a Cam Profile, the instruction produces an error.	
		Value	**Description**
		0	Cam profile element has not been calculated.
		1	Cam profile element is being calculated.
		2	Cam profile element has been calculated.
		n	Cam profile element has been calculated and is currently being used by (n-2) MAPC and MATC instructions.

MOTION_GROUP Structure

There is one MOTION_GROUP data type per controller. This structure contains status and configuration information about the motion group.

Table 161 - MOTION_GROUP Status and Configuration Descriptions

Enumerations	Data Type	Description			
GroupStatus	DINT	The status bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		InhibitStatus	00	DINT	Inhibit status.
		GroupSynced	01	DINT	Synchronization status. Group sync status is only cleared on a group overlap or CST loss fault.
		AxisInhibitStatus	03	DINT	
		-no-tag	02	DINT	Timer Event started.
		Reserved	03…31		
MotionFault	DINT	The motion fault bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		ACAsyncConnFault	00	DINT	Asynchronous connection fault.
		ACSyncConnFault	01	DINT	Synchronous connection fault.
		Reserved	02…31		

Table 161 - MOTION_GROUP Status and Configuration Descriptions (Continued)

Enumerations	Data Type	Description			
ServoFault	DINT	The servo-module fault bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		POtrvlFault	00	DINT	Positive overtravel fault.
		NOtrvlFault	01	DINT	Negative overtravel fault.
		PosErrorFault	02	DINT	Position error fault.
		EncCHALossFault	03	DINT	Encoder channel A loss fault.
		EncCHBLossFault	04	DINT	Encoder channel B loss fault.
		EncCHZLossFault	05	DINT	Encoder channel Z loss fault.
		EncNsFault	06	DINT	Encoder noise fault.
		DriveFault	07	DINT	Drive fault.
		Reserved	08…31		
		Bit	**Number**	**Data Type**	**Description**
		SyncConnFault	00	DINT	Synchronous connection fault.
		HardFault	01	DINT	Servo hardware fault.
		Reserved	02…31		
GroupFault	DINT	The fault bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		GroupOverlapFault	00	DINT	Group task overlap fault.
		CSTLossFault	01	DINT	The controller has lost synchronization with the CST master.
		GroupTaskLoadingFault	02	DINT	The group coarse update period is too low, user application tasks are not getting enough time to execute.
		Reserved	03…31		
AxisFault	DINT	The fault bits for the axis			
		Bit	**Number**	**Data Type**	**Description**
		PhysicalAxisFault	00	DINT	A Servo or Drive fault has occurred.
		ModuleFault	01	DINT	A serious fault has occurred with the motion module associated with the selected axis. Usually affects all axes associated with the motion module.
		ConfigFault	02	DINT	One or more axis attributes associated with a motion module or drive has not been successfully updated to match the value of the corresponding attribute of the local controller.
		Reserved	03…31		

MOTION_INSTRUCTION Data Type

You must define a motion control tag for each motion instruction that you use. The tag uses the MOTION_INSTRUCTION data type and stores status information about the instruction.

Table 162 - MOTION_INSTRUCTION Data Type Descriptions

Enumerations	Data Type	Description	
FLAGS	DINT	Use this DINT to access all the status bits for the instruction in one 32-bit value.	
		For this status bit	**Use this bit number**
		EN	31
		DN	29
		ER	28
		PC	27
		IP	26
		AC	23
		DECEL	1
		ACCEL	0
EN	BOOL	The enable bit indicates that the instruction is enabled (the rung-in and rung-out condition is true).	
DN	BOOL	The done bit indicates that all calculations and messaging (if any) are complete.	
ER	BOOL	The error bit indicates when the instruction is used illegally.	
PC	BOOL	The process complete bit indicates that the operation is complete. The .DN bit sets after an instruction has completed execution. The .PC bit sets when the initiated process has completed.	
IP	BOOL	The in process bit indicates that a process is being executed.	
AC	BOOL	The Active Bit lets you know which instruction is controlling the motion when you have instructions queued. It sets when the instruction becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.	
ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing	
DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.	
ERR	INT	The error value contains the error code associated with a motion function. See Error Codes (ERR) for Motion Instructions on page 345.	
STATUS	SINT	The status of any message associated with the motion function.	
		Message Status	**Description**
		0x0	The message was successful.
		0x1	The module is processing another message.
		0x2	The module is waiting for a response to a previous message.
		0x3	The response to a message failed.
		0x4	The module is not ready for messaging.
STATE	SINT	The execution status value keeps track of the execution state of a function. Many motion functions have several steps and this value tracks these steps. The execution status is always set to 0 when the controller sets the EN bit for a motion instruction. Other execution states depend on the motion instruction.	
SEGMENT	DINT	A segment is the distance from one point up to but, not including the next point. A SEGMENT gives the relative position by segment number as the Cam is executing.	
EXERR	SINT	Extended error code —use it for more information about an error.	

OUTPUT_CAM Structure

The OUTPUT_CAM data type is an array that defines the specifics for each Output Cam element. The OUTPUT_CAM contains the following members.

Table 163 - OUTPUT_CAM Member Descriptions

Enumerations	Data Type	Description	
OutputBit	DINT	You must select an output bit within the range of 0 to 31. A selection of less than 0 or greater than 31 results in an Illegal Output Cam error and the cam element is not considered.	
LatchType	DINT	The Latch Type determines how the corresponding output bit is set. A value of less than 0 or greater than 3 results in an Illegal Output Cam error and a latch type of Inactive is used.	
		Value	**Description**
		0 = Inactive	The output bit is not changed.
		1 = Position	The output bit is set when the axis enters the compensated cam range.
		2 = Enable	The output bit is set when the enable bit becomes active.
		3 = Position and Enable	The output bit is set when the axis enters the compensated cam range and the enable bit becomes active.
UnlatchType	DINT	The Unlatch Type determines how the output bit is reset. Selecting a value less than 0 or greater than 5 results in an Illegal Output Cam error and an unlatch type of Inactive is used.	
		Value	**Description**
		0 = Inactive	The output bit is not changed.
		1 = Position	The output bit is reset when the axis leaves the compensated cam range.
		2 = Duration	The output bit is reset when the duration expires.
		3 = Enable	The output bit is reset when the enable bit becomes inactive.
		4 = Position and Enable	The output bit is reset when the axis leaves the compensated cam range or the enable bit becomes inactive.
		5 = Duration and Enable	The output bit is reset when the duration expires or the enable bit becomes inactive.
Left	REAL	The left cam position along with the right cam position define the cam range of the Output Cam element. The left and right cam positions specify the latch or unlatch positions of the output bit when the latch or unlatch type is set to **Position** or **Position and Enable** with the enable bit active. If the left position is less than the Cam Start position or greater than the Cam End position, an Illegal Output Cam error is returned and the cam element is not considered.	
Right	REAL	The right cam position along with the left cam position define the cam range of the Output Cam element. The right and left cam positions specify the latch or unlatch positions of the output bit when the latch or unlatch type is set to **Position** or **Position and Enable** with the enable bit active. If the right position is less than the Cam Start position or greater than the Cam End position, an Illegal Output Cam error is returned and the cam element is not considered.	
Duration	REAL	Duration specifies the time in seconds between latching and unlatching when the Unlatch Type is **Duration** or **Duration and Enable** with the enable bit active. A value less than or equal to 0 results in an Illegal Output Cam error and the cam element is not considered.	
EnableType	DINT	This defines the source and polarity of the specified Enable bit when LatchType or UnlatchType is **Enable**, **Position and Enable** or **Duration and Enable**. A value of less than 0 or greater than 31 results in an Illegal Output Cam error and the cam element is not considered.	
		Value	**Description**
		0 = Input	The enable bit is in the Input parameter.
		1 = Inverted Input	The enable bit is in the input parameter and is active low.
		2 = Output	The enable bit is in the Output parameter.
		3 = Inverted Output	The enable bit is in the Output parameter and is active low.
EnableBit	DINT	The value of the Enable Bit selected must be between 0 and 31 when LatchType or UnlatchType is **Enable**, **Position and Enable** or **Duration and Enable**. A value of less than 0 or greater than 31 results in an Illegal Output Cam error and the cam element is not considered.	

OUTPUT_COMPENSATION Structure

The OUTPUT_COMPENSATION data type defines the details for each output bit by setting the characteristics of each actuator. OUTPUT_COMPENSATION contains the following members.

Table 164 - OUTPUT_COMPENSATION Member Descriptions

Enumerations	Data Type	Description	
Offset	REAL	Offset provides position compensation for both the latch and unlatch operations.	
LatchDelay	REAL	Latch delay, programmed in seconds, provides time compensation for the latch operation.	
UnlatchDelay	REAL	Unlatch delay, programmed in seconds, provides time compensation for the unlatch operation.	
Mode	DINT	The Mode determines the behavior of the output bit. The following four mode options are available. A value of less than 0 or greater than 3 results in an Illegal Output Compensation error.	
		Value	**Description**
		0 = Normal	The output bit is set for the latch operation and is reset for the unlatch operation.
		1 = Inverted	The output bit is reset for the latch operation and is set for the unlatch the operation.
		2 = Pulsed	The output bit is set for the latch operation and for the on-duty state of pulse and is reset for the unlatch operation and for the off-duty state of the pulse.
		3 = Inverted and Pulsed	The output bit is reset for the latch operation and for the on-duty state of the pulse and is set for the unlatch operation and for the off-duty state of the pulse.
CycleTime	REAL	Pulse time in seconds. If mode is **Pulsed** or **Inverted and Pulsed**, and CycleTime is less than or equal to 0, an Illegal Output Compensation error results.	
DutyCycle	REAL	The percent of CycleTime that the pulse is to be turned on (on-duty). A value of 50 represents 50% on-duty. A value of less than 0 or greater than 100 returns an Illegal Output Compensation error.	

Structured Text Programming

This appendix describes issues that are unique with structured text programming. Review the information in this appendix to make sure you understand how your structured text programming will execute.

Topic	Page
Structured Text Syntax	359
Assignments	360
Expressions	362
Instructions	369
Constructs	370
Comments	387

Structured Text Syntax

Structured text is a textual programming language that uses statements to define what to execute.

- Structured text is not case sensitive.
- Use tabs and carriage returns (separate lines) to make your structured text easier to read. They have no effect on the execution of the structured text.

Structured text is not case sensitive. Structured text can contain these components.

Table 165 - Structured Text Component Descriptions

Term	Descriptions		Examples
Assignment (see page 360)	Use an assignment statement to assign values to tags. The := operator is the assignment operator. Terminate the assignment with a semi colon ";".		tag:= expression;
Expression (see page 362)	An expression is part of a complete assignment or construct statement. An expression evaluates to a number (numerical expression) or to a true or false state (BOOL expression). An expression contains the following.		
	Tags	A named area of the memory where data is stored (BOOL, SINT,INT,DINT, REAL, string).	value1
	Immediate	A constant value.	4
	Operators	A symbol or mnemonic that specifies an operation within an expression.	tag1 + tag2 tag1 >= value1
	Functions	When executed, a function yields one value. Use parentheses to contain the operand of a function. Even though their syntax is similar, functions differ from instructions in that functions can only be used in expressions. Instructions cannot be used in expressions.	function(tag1)

Table 165 - Structured Text Component Descriptions (Continued)

Term	Descriptions	Examples
Instruction (see page 369)	An instruction is a standalone statement. An instruction uses parenthesis to contain its operands. Depending on the instruction, there can be zero, one, or multiple operands. When executed, an instruction yields one or more values that are part of a data structure. Terminate the instruction with a semi colon ";". Even though their syntax is similar, instructions differ from functions in that instructions cannot be used in expressions. Functions can only be used in expressions.	instruction(); *instruction(operand);* *instruction(operand1, operand2,operand3);*
Construct (see page 370)	A conditional statement used to trigger structured text code (i.e, other statements). Terminate the construct with a semi colon ";".	IF...THEN CASE FOR...DO WHILE...DO REPEAT...UNTIL EXIT
Comment (see page 387)	Text that explains or clarifies what a section of structured text does. • Use comments to make it easier to interpret the structured text. • Comments do not affect the execution of the structured text. • Comments can appear anywhere in structured text.	*//comment* (**start of comment . . . end of comment**) */*start of comment . . . end of comment*/*

Assignments

Use an assignment to change the value stored within a tag. An assignment has this syntax:

tag := *expression* ;

Table 166 - Syntax Descriptions

Component	Description	
tag	represents the tag that is getting the new value the tag must be a BOOL, SINT, INT, DINT, or REAL	
:=	is the assignment symbol	
expression	represents the new value to assign to the tag	
	If *Tag* is This Data Type	**Use This Type of Expression**
	BOOL	BOOL expression
	SINT INT DINT REAL	numeric expression
;	ends the assignment	

The *tag* retains the assigned value until another assignment changes the value.

The expression can be simple, such as an immediate value or another tag name, or the expression can be complex and include several operators and/or functions. For more information, see Expressions on page 362.

TIP I/O module data updates asynchronously to the execution of logic. If you reference an input multiple times in your logic, the input could change state between separate references. If you need the input to have the same state for each reference, buffer the input value and reference that buffer tag.

For more information, see Logix5000 Controllers Common Procedures, publication 1756-PM001.

Specify a Non-retentive Assignment

The non-retentive assignment is different from the regular assignment described in Assignments on page 360 in that the tag in a non-retentive assignment is reset to zero each time the controller:

- enters the RUN mode.
- leaves the step of an SFC if you configure the SFC for *Automatic reset*. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

A non-retentive assignment has this syntax:

tag [:=] *expression* ;

Table 167 - Syntax Descriptions

Component	Description	
tag	represents the tag that is getting the new value the tag must be a BOOL, SINT, INT, DINT, or REAL	
[:=]	is the non-retentive assignment symbol	
expression	represents the new value to assign to the tag	
	If Tag is This Data Type	**Use This Type of Expression**
	BOOL	BOOL expression
	SINT INT DINT REAL	numeric expression
;	ends the assignment	

Assign an ASCII Character to a String

Use the assignment operator to assign an ASCII character to an element of the DATA member of a string tag. To assign a character, specify the value of the character or specify the tag name, DATA member, and element of the character. These are some examples.

This is OK	This is Not OK.
string1.DATA[0]:= 65;	string1.DATA[0] := A;
string1.DATA[0]:= string2.DATA[0];	string1 := string2;

To add or insert a string of characters to a string tag, use either of these ASCII string instructions.

To	Use This Instruction
add characters to the end of a string	CONCAT
insert characters into a string	INSERT

Expressions

An expression is a tag name, equation, or comparison. To write an expression, use any of these elements:

- Tag name that stores the value (variable)
- Number that you enter directly into the expression (immediate value)
- Functions, such as, ABS, TRUNC
- Operators, such as, +, -, <, >, And, Or

As you write expressions, follow these general rules:

- Use any combination of upper-case and lower-case letter. For example, these three variations of 'AND' are acceptable: AND, And, and.
- For more complex requirements, use parentheses to group expressions within expressions. This makes the whole expression easier to read and ensures that the expression executes in the desired sequence.

  For more information, see Determine the Order of Execution on page 368.

In structured text, you use these two types of expressions:

BOOL expression: An expression that produces either the BOOL value of 1 (true) or 0 (false).

- A boolean expression uses BOOL tags, relational operators, and logical operators to compare values or check if conditions are true or false. For example, tag1>65.
- A simple BOOL expression can be a single BOOL tag.
- Typically, you use BOOL expressions to condition the execution of other logic.

Numeric expression: An expression that calculates an integer or floating-point value.

- A numeric expression uses arithmetic operators, arithmetic functions, and bitwise operators. For example, tag1+5.
- Often, you nest a numeric expression within a bool expression. For example, (tag1+5)>65.

Use the following table to choose operators for your expressions.

Table 168 - Choose Expression Operators

If You Want To	Then See
Calculate an arithmetic value	Use Arithmetic Operators and Functions on page 363.
Compare two values or strings	Use Relational Operatorson page 365.
Check if conditions are true or false	Use Logical Operators on page 366.
Compare the bits within values	Use Bitwise Operators on page 367.

Use Arithmetic Operators and Functions

You can combine multiple operators and functions in arithmetic expressions.

Table 169 - Arithmetic Operators Calculate Values

To	Use this Operator	Optimal Data Type
add	+	DINT, REAL
subtract/negate	-	DINT, REAL
multiply	*	DINT, REAL
exponent (x to the power of y)	**	DINT, REAL
divide	/	DINT, REAL
modulo-divide	MOD	DINT, REAL

Arithmetic functions perform math operations. Specify a constant, a non-boolean tag, or an expression for the function.

Table 170 - Arithmetic Functions

For	Use this function	Optimal Data Type
absolute value	ABS (*numeric_expression*)	DINT, REAL
arc cosine	ACOS (*numeric_expression*)	REAL
arc sine	ASIN (*numeric_expression*)	REAL
arc tangent	ATAN (*numeric_expression*)	REAL
cosine	COS (*numeric_expression*)	REAL
radians to degrees	DEG (*numeric_expression)*	DINT, REAL
natural log	LN (*numeric_expression*)	REAL
log base 10	LOG (*numeric_expression*)	REAL
degrees to radians	RAD (*numeric_expression*)	DINT, REAL
sine	SIN (*numeric_expression*)	REAL
square root	SQRT (*numeric_expression*)	DINT, REAL
tangent	TAN (*numeric_expression*)	REAL
truncate	TRUNC (*numeric_expression*)	DINT, REAL

These are some examples.

Table 171 - Arithmetic Functions Example Situations

Use This Format	Example	
	For this Situation	**You'd Write**
value1 operator value2	If gain_4 and gain_4_adj are DINT tags and your specification says: 'Add 15 to gain_4 and store the result in gain_4_adj.'	gain_4_adj := gain_4+15;
operator value1	If alarm and high_alarm are DINT tags and your specification says: "Negate high_alarm and store the result in alarm."	alarm:= -high_alarm;
function(numeric_expression)	If overtravel and overtravel_POS are DINT tags and your specification says: "Calculate the absolute value of overtravel and store the result in overtravel_POS."	overtravel_POS := ABS(overtravel);
value1 operator (function((value2+value3)/2)	If adjustment and position are DINT tags and sensor1 and sensor2 are REAL tags and your specification says: "Find the absolute value of the average of sensor1 and sensor2, add the adjustment, and store the result in position."	position := adjustment + ABS((sensor1 + sensor2)/2);

Use Relational Operators

Relational operators compare two values or strings to provide a true or false result. The result of a relational operation is a BOOL value.

If the Comparison Is	The Result Is
true	1
false	0

Use these relational operators.

For This Comparison	Use This Operator	Optimal Data Type
equal	=	DINT, REAL, string
less than	<	DINT, REAL, string
less than or equal	<=	DINT, REAL, string
greater than	>	DINT, REAL, string
greater than or equal	>=	DINT, REAL, string
not equal	<>	DINT, REAL, string

These are some examples.

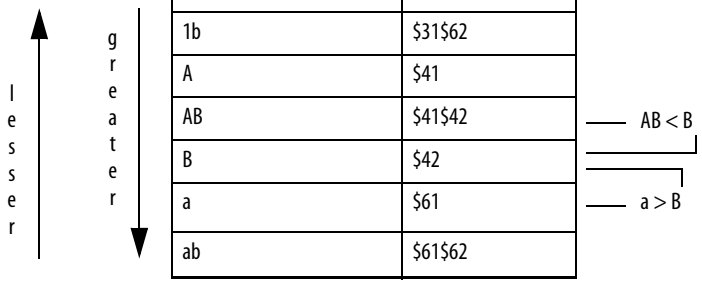
Table 172 - Relational Operators Example Situations

Use This Format	Example	
	For This Situation	**You'd Write**
value1 operator value2	If temp is a DINT tag and your specification says: "If temp is less than 100· then…"	IF temp<100 THEN...
stringtag1 operator stringtag2	If bar_code and dest are string tags and your specification says: "If bar_code equals dest then…"	IF bar_code=dest THEN...
char1 operator char2 To enter an ASCII character directly into the expression, enter the decimal value of the character.	If bar_code is a string tag and your specification says: "If bar_code.DATA[0] equals 'A' then…"	IF bar_code.DATA[0]=65 THEN...
bool_tag := *bool_expressions*	If count and length are DINT tags, done is a BOOL tag, and your specification says "If count is greater than or equal to length, you are done counting."	done := (count >= length);

How Strings are Evaluated

The hexadecimal values of the ASCII characters determine if one string is less than or greater than another string.

- When the two strings are sorted as in a telephone directory, the order of the strings determines which one is greater.



ASCII Characters	Hex Codes
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

- Strings are equal if their characters match.
- Characters are case sensitive. Upper case 'A' ($41) is *not* equal to lower case 'a' ($61).

Use Logical Operators

Logical operators let you check if multiple conditions are true or false. The result of a logical operation is a BOOL value.

If The Comparison is	The Result is
true	1
false	0

Use these logical operators.

Table 173 - Logical operators

For	Use this Operator	Data Type
logical AND	&, AND	BOOL
logical OR	OR	BOOL
logical exclusive OR	XOR	BOOL
logical complement	NOT	BOOL

These are some examples.

Table 174 - Logical Operator Example Situations

Use this format	Example	
	For This Situation	**You'd Write**
BOOLtag	If photoeye is a BOOL tag and your specification says: `If photoeye_1 is on then…'	IF photoeye THEN...
NOT *BOOLtag*	If photoeye is a BOOL tag and your specification says: `If photoeye is off then…'	IF NOT photoeye THEN...
expression1 & *expression2*	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: `If photoeye is on and temp is less than 100· then…'	IF photoeye & (temp<100) THEN...
expression1 OR *expression2*	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: `If photoeye is on or temp is less than 100· then…'	IF photoeye OR (temp<100) THEN...
expression1 XOR *expression2*	If photoeye1 and photoeye2 are BOOL tags and your specification says: `If: • photoeye1 is on while photoeye2 is off or • photoeye1 is off while photoeye2 is on then…'	IF photoeye1 XOR photoeye2 THEN...
BOOLtag := *expression1* & *expression2*	If photoeye1 and photoeye2 are BOOL tags, open is a BOOL tag, and your specification says: `If photoeye1 and photoeye2 are both on, set open to true'.	open := photoeye1 & photoeye2;

Use Bitwise Operators

Bitwise operators manipulate the bits within a value based on two values.

Table 175 - Bitwise Operators

For	Use this Operator	Optimal Data Type
bitwise AND	&, AND	DINT
bitwise OR	OR	DINT
bitwise exclusive OR	XOR	DINT
bitwise complement	NOT	DINT

These are some examples.

Table 176 - Bitwise Operator Example Situation

Use this Format	Example	
	For this Situation	**You'd Write**
value1 operator value2	If input1, input2, and result1 are DINT tags and your specification says: "Calculate the bitwise result of input1 and input2. Store the result in result1."	result1 := input1 AND input2;

Determine the Order of Execution

The operations you write into an expression are performed in a prescribed order, not necessarily from left to right.

- Operations of equal order are performed from left to right.
- If an expression contains multiple operators or functions, group the conditions in parenthesis '( )'. This ensures the correct order of execution and makes it easier to read the expression.

Table 177 - Operand Order of Execution

Order	Operation
1	( )
2.	function (. . .)
3.	**
4.	– (negate)
5.	NOT
6.	*, /, MOD
7.	+, - (subtract)
8.	<, <=, >, >=
9.	=, <>
10.	&, AND
11.	XOR
12.	OR

Instructions

Structured text statements can also be instructions. See the Locator Table at the beginning of this manual for a list of the instructions available in structured text. A structured text instruction executes each time it is scanned. A structured text instruction within a construct executes every time the conditions of the construct are true. If the conditions of the construct are false, the statements within the construct are not scanned. There is no rung-condition or state transition that triggers execution.

This differs from function block instructions that use EnableIn to trigger execution. Structured text instructions execute as if EnableIn is always set.

This also differs from relay ladder instructions that use rung-condition-in to trigger execution. Some relay ladder instructions only execute when rung-condition-in toggles from false to true. These are transitional relay ladder instructions. In structured text, instructions will execute each time they are scanned unless you precondition the execution of the structured text instruction.

For example, the ABL instruction is a transitional instruction in relay ladder. In this example, the ABL instruction only executes on a scan when tag_xic transitions from cleared to set. The ABL instruction does not execute when tag_xic stays set or when tag_xic is cleared.

The ABL instruction will execute every scan that tag_zic is set, not just when tag_xic transition from clear to set in structured text if you write this example as follows.

```
IF tag_xic THEN ABL(0,serial_control);

END_IF;
```

the ABL instruction will execute every scan that tag_xic is set, not just when tag_xic transitions from cleared to set.

If you want the ABL instruction to execute only when tag_xic transitions from cleared to set, you have to condition the structured text instruction. Use a one shot to trigger execution.

```
osri_1.InputBit := tag_xic;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
        ABL(0,serial_control);
END_IF;
```

Constructs

Constructs can be programmed singly or nested within other constructs.

Table 178 - Programming Constructs

If You Want To	Use This Construct	Available in These Languages	Page
do something if or when specific conditions occur	IF...THEN	structured text	371
select what to do based on a numerical value	CASE...OF	structured text	375
do something a specific number of times before doing anything else	FOR...DO	structured text	378
keep doing something as long as certain conditions are true	WHILE...DO	structured text	381
keep doing something until a condition is true	REPEAT...UNTIL	structured text	384

Key Words Reserved for Future Use

These constructs are not available:

- GOTO
- REPEAT

Logix Designer programming software does not let you use them.

IF...THEN

Use IF...THEN to do something if or when specific conditions occur.

Operands

Structured Text

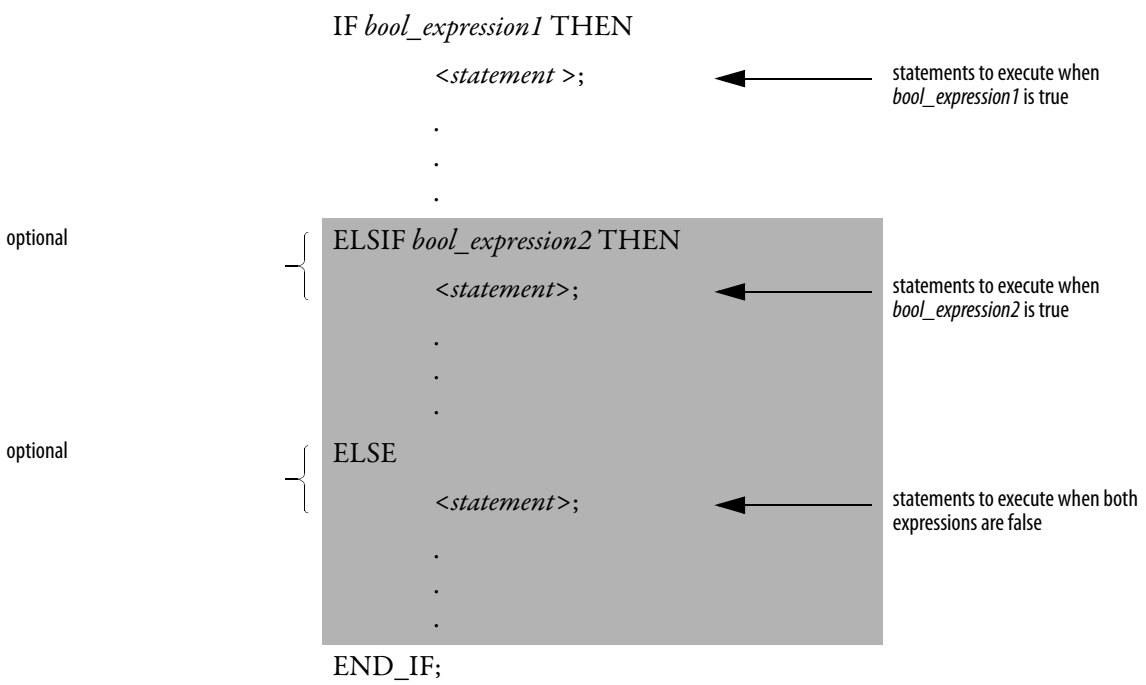
IF *bool_expression* THEN<*statement*>;END_IF;

Table 179 - IF...THEN Operand Description

Operand	Type	Format	Enter
bool_expression	BOOL	Tag Expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

Description

The syntax is as follows.



To use ELSIF or ELSE, follow these guidelines.

1. To select from several possible groups of statements, add one or more ELSIF statements.
   - Each ELSIF represents an alternative path.
   - Specify as many ELSIF paths as you need.
   - The controller executes the first true IF or ELSIF and skips the rest of the ELSIFs and the ELSE.

2. To do something when all of the IF or ELSIF conditions are false, add an ELSE statement.

This table summarizes combinations of IF, THEN, ELSIF, and ELSE.

If You Want To	And	Then Use This Construct
do something if or when conditions are true	do nothing if conditions are false	IF…THEN
	do something else if conditions are false	IF…THEN…ELSE
choose from alternative statements (or groups of statements) based on input conditions	do nothing if conditions are false	IF…THEN…ELSIF
	assign default statements if all conditions are false	IF…THEN…ELSIF…ELSE

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

Example 1

IF...THEN

If You Want This	Enter This Structured Text
IF rejects > 3 then	IF rejects > 3 THEN
conveyor = off (0)	conveyor := 0;
alarm = on (1)	alarm := 1;
	END_IF;

Example 2

IF...THEN...ELSE

If You Want This	Enter This Structured Text
If conveyor direction contact = forward (1) then	IF conveyor_direction THEN
light = off	light := 0;
Otherwise light = on	ELSE
	light [:=] 1;
	END_IF;

The [:=] tells the controller to clear light whenever the controller:

- enters the RUN mode.
- leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 3

IF…THEN…ELSIF

If You Want This	Enter This Structured Text
If sugar low limit switch = low (on) and sugar high limit switch = not high (on) then	IF Sugar.Low & Sugar.High THEN
inlet valve = open (on)	Sugar.Inlet [:=] 1;
Until sugar high limit switch = high (off)	ELSIF NOT(Sugar.High) THEN
	Sugar.Inlet := 0;
	END_IF;

The [:=] tells the controller to clear Sugar.Inlet whenever the controller:

- enters the RUN mode
- leaves the step of an SFC if you configure the SFC for Automatic reset (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 4

IF…THEN…ELSIF…ELSE

If You Want This	Enter This Structured Text
If tank temperature > 100	IF tank.temp > 200 THEN
then pump = slow	pump.fast :=1; pump.slow :=0; pump.off :=0;
If tank temperature > 200	ELSIF tank.temp > 100 THEN
then pump = fast	pump.fast :=0; pump.slow :=1; pump.off :=0;
otherwise pump = off	ELSE
	pump.fast :=0; pump.slow :=0; pump.off :=1;
	END_IF;

CASE...OF

Use CASE to select what to do based on a numerical value.

Operands

Structured Text

CASE *numeric_expression* OF

selector1: *statement*;

selectorN: *statement*;

ELSE

statement;

END_CASE;

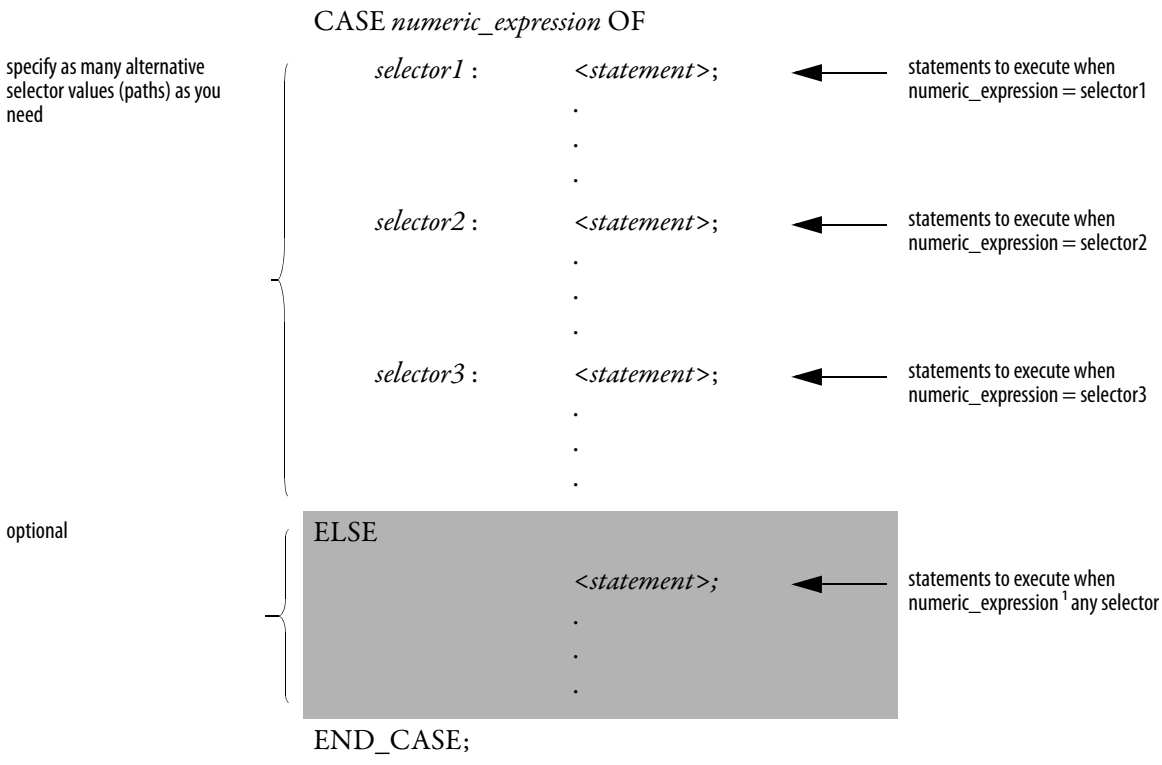
IMPORTANT If you use REAL values, use a range of values for a selector because a REAL value is more likely to be within a range of values than an exact match of one, specific value.

Table 180 - CASE...OF Operand Descriptions

Operand	Type	Format	Enter
numeric_expression	SINT INT DINT REAL	Tag Expression	tag or expression that evaluates to a number (numeric expression)
selector	SINT INT DINT REAL	Immediate	same type as *numeric_expression*

Description

The syntax is as follows.



This is the syntax for entering the selector values.

Table 181 - Selector Values Syntax

When Selector Is	Enter
one value	value: statement
multiple, distinct values	value1, value2, valueN : <statement> Use a comma (,) to separate each value.
a range of values	value1...valueN : <statement> Use two periods (..) to identify the range.
distinct values plus a range of values	valuea, valueb, value1...valueN : <statement>

The CASE construct is similar to a switch statement in the C or C++ programming languages. However, with the CASE construct the controller executes only the statements that are associated with the first matching selector value. Execution always breaks after the statements of that selector and goes to the END_CASE statement.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

There are no fault conditions.

Example

Table 182 - CASE...OF Structured Text Example

If You Want This	Enter This Structured Text
If recipe number = 1 then	CASE recipe_number OF
Ingredient A outlet 1 = open (1)	1: Ingredient_A.Outlet_1 :=1;
Ingredient B outlet 4 = open (1)	Ingredient_B.Outlet_4 :=1;
If recipe number = 2 or 3 then	2,3: Ingredient_A.Outlet_4 :=1;
Ingredient A outlet 4 = open (1)	Ingredient_B.Outlet_2 :=1;
Ingredient B outlet 2 = open (1)	
If recipe number = 4, 5, 6, or 7 then	4…7: Ingredient_A.Outlet_4 :=1;
Ingredient A outlet 4 = open (1)	Ingredient_B.Outlet_2 :=1;
Ingredient B outlet 2 = open (1)	
If recipe number = 8, 11, 12, or 13 then	8, 11…13 Ingredient_A.Outlet_1 :=1;
Ingredient A outlet 1 = open (1)	Ingredient_B.Outlet_4 :=1;
Ingredient B outlet 4 = open (1)	
Otherwise all outlets = closed (0)	ELSE
	Ingredient_A.Outlet_1 [:=]0;
	Ingredient_A.Outlet_4 [:=]0;
	Ingredient_B.Outlet_2 [:=]0;
	Ingredient_B.Outlet_4 [:=]0;
	END_CASE;

The [:=] tells the controller to also clear the outlet tags whenever the controller:

- Enters the RUN mode.
- Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

FOR...DO

Use the FOR...DO loop to do something a specific number of times before doing anything else.

Operands

Structured Text

FOR *count*:= *initial_value* TO *final_value* BY *increment* DO

  <*statement*>;

END_FOR;

IMPORTANT Make sure that you do not iterate within the loop too many times in a single scan.

- The controller does not execute any other statements in the routine until it completes the loop.
- If the time that it takes to complete the loop is greater than the watchdog timer for the task, a major fault occurs.
- Consider using a different construct, such as IF...THEN.

Table 183 - FOR...DO Operand Descriptions

Operand	Type	Format	Description
count	SINT INT DINT	Tag	Tag to store count position as the FOR. . . DO executes
initial_ value	SINT INT DINT	Tag Expression Immediate	must evaluate to a number specifies initial value for count
final_ value	SINT INT DINT	Tag Expression Immediate	specifies final value for count, that determines when to exit the loop
increment	SINT INT DINT	Tag Expression Immediate	(*optional*) amount to increment count each time through the loop If you don't specify an increment, the count increments by 1.

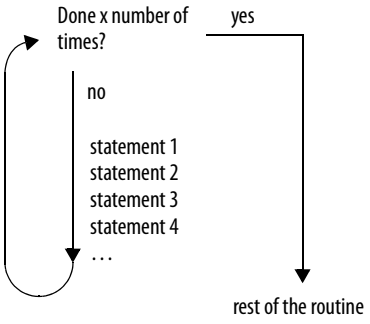
Description

This is the syntax.

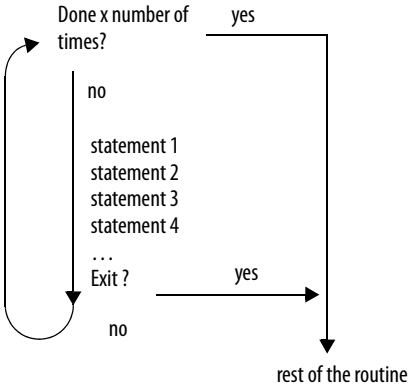
Table 184 - FOR...DO Syntax Description

		FOR *count*:= *initial_value*	
		TO *final_value*	
optional	{	BY *increment*	If you don't specify an increment, the loop increments by 1.
		DO	
		<*statement*>;	
optional	{	IF *bool_expression* THEN	
		EXIT;	If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.
		END_IF;	
		END_FOR;	

These diagrams show how a FOR...DO loop executes and how an EXIT statement leaves the loop early.



The FOR…DO loop executes a specific number of times.



To stop the loop before the count reaches the last value, use an EXIT statement.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

A major fault will occur if the construct loops are too long. The fault type is 6 and the fault code is 1.

Example 1

If you want this	Enter this structured text

Clear bits 0 to 31 in an array of BOOL's.
1. Initialize the subscript tag to 0.
2. Clear array[subscript] . For example, when subscript = 5, clear array[5].
3. Add 1 to subscript.
4. If subscript is £ to 31, repeat 2 and 3.
Otherwise, stop.
	For subscript:=0 to 31 by 1 do
	  array[subscript]:= 0;
	End_for;

Example 2

If You Want This	Enter This Structured Text

A user-defined data type (structure) stores this information about an item in your inventory:
• Barcode ID of the item (string data type)
• Quantity in stock of the item (DINT data type)
An array of the previously listed structure contains an element for each different item in your inventory. You want to search the array for a specific product (use its bar code) and determine the quantity that is in stock.
1. Get the size (number of items) of the Inventory array and store the result in Inventory_Items (DINT tag).
2. Initialize the position tag to 0.
3. If Barcode matches the ID of an item in the array, then:
a. Set the Quantity tag = Inventory[position].Qty. This produces the quantity in stock of the item.
b. Stop.
Barcode is a string tag that stores the bar code of the item that you are searching. For example, when position = 5, compare Barcode to Inventory[5].ID.
4. Add 1 to position.
5. If position is £ to (Inventory_Items -1), repeat 3 and 4. Because element numbers start at 0, the last element is 1 less than the number of elements in the array.
Otherwise, stop.
	SIZE(Inventory,0,Inventory_Items);
	For position:=0 to Inventory_Items - 1 do
	  If Barcode = Inventory[position].ID then
	  Quantity := Inventory[position].Qty;
	  Exit;
	  End_if;
	End_for;

WHILE...DO

Use the WHILE...DO loop to keep doing something as long as certain conditions are true.

Operands

Structured Text

WHILE *bool_expression* DO

  <*statement*>;

END_WHILE;

IMPORTANT Make sure that you do not iterate within the loop too many times in a single scan.

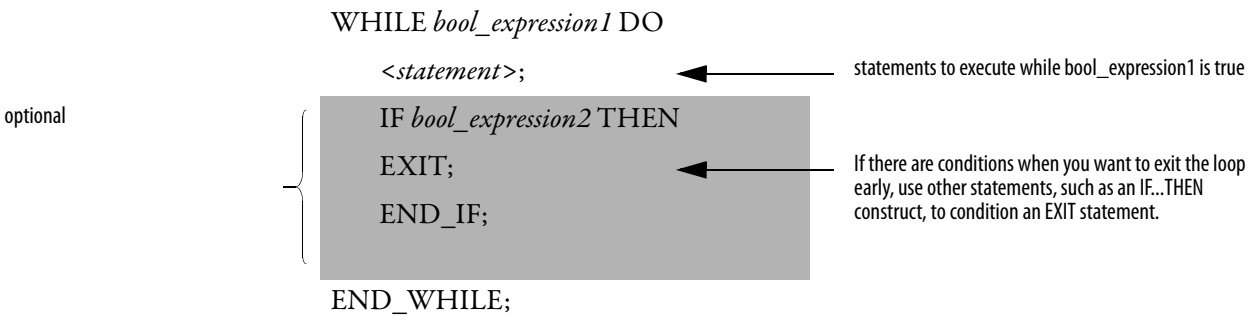
- The controller does not execute any other statements in the routine until it completes the loop.
- If the time that it takes to complete the loop is greater than the watchdog timer for the task, a major fault occurs.
- Consider using a different construct, such as IF...THEN.

Table 185 - WHILE...DO Operand Description

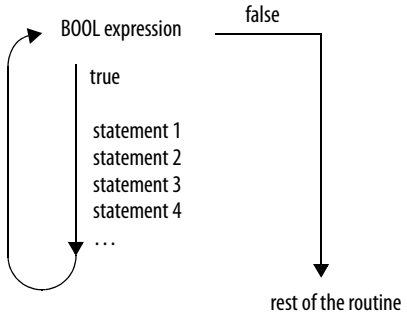
Operand	Type	Format	Enter
bool_expression	BOOL	Tag Expression	BOOL tag or expression that evaluates to a BOOL value

Description

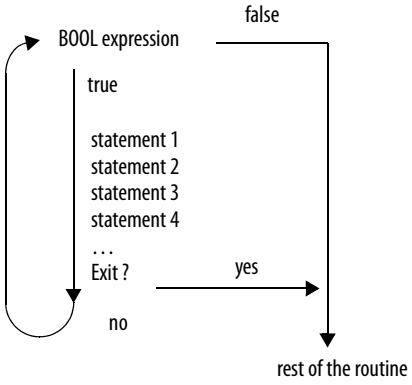
This is the syntax.


WHILE bool_expression1 DO
<statement>;
statements to execute while bool_expression1 is true
optional
IF bool_expression2 THEN
EXIT;
If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.
END_IF;
END_WHILE;

These diagrams show how a WHILE...DO loop executes and how an EXIT statement leaves the loop early.



While the *bool_expression* is true, the controller executes only the statements within the WHILE…DO loop.



To stop the loop before the conditions are true, use an EXIT statement.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

A major fault will occur if the construct loops are too long. The fault type is 6 and the fault code is 1.

Example 1

If You Want This	Enter This Structured Text

The WHILE...DO loop evaluates its conditions first. If the conditions are true, the controller then executes the statements within the loop.
This differs from the REPEAT...UNTIL loop because the REPEAT...UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again. The statements in a REPEAT...UNTIL loop are always executed at least once. The statements in a WHILE...DO loop might never be executed.
	pos := 0;
	While ((pos <= 100) & structarray[pos].value <> targetvalue)) do
	  pos := pos + 2;
	  String_tag.DATA[pos] := SINT_array[pos];
	end_while;

Example 2

If You Want This	Enter This Structured Text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
1. Initialize Element_number to 0.
2. Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
3. If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.
4. Set String_tag[element_number] = the character at SINT_array[element_number].
5. Add 1 to element_number. This lets the controller check the next character in SINT_array.
6. Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.)
7. If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.)
8. Click step 3.
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	While SINT_array[element_number] <> 13 do
	  String_tag.DATA[element_number] := SINT_array[element_number];
	  element_number := element_number + 1;
	  String_tag.LEN := element_number;
	  If element_number = SINT_array_size then
	    exit;
	  end_if;
	end_while;

REPEAT...UNTIL

Use the REPEAT...UNTIL loop to keep doing something until conditions are true.

Operands

Structured Text

REPEAT

<*statement*>;

UNTIL *bool_expression*

END_REPEAT;

IMPORTANT Make sure that you do not iterate within the loop too many times in a single scan.

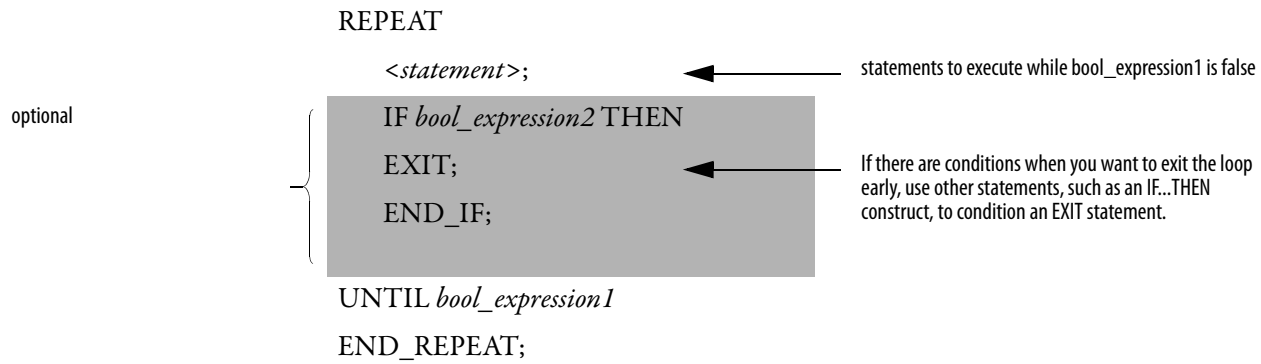
- The controller does not execute any other statements in the routine until it completes the loop.
- If the time that it takes to complete the loop is greater than the watchdog timer for the task, a major fault occurs.
- Consider using a different construct, such as IF...THEN.

Table 186 - REPEAT...UNTIL Operand Description

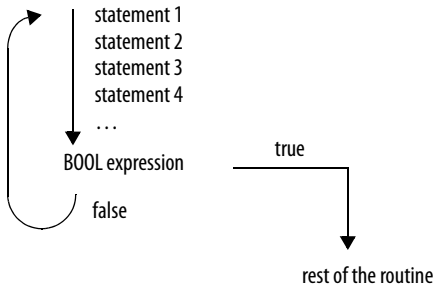
Operand	Type	Format	Enter
bool_expression	BOOL	Tag Expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

Description

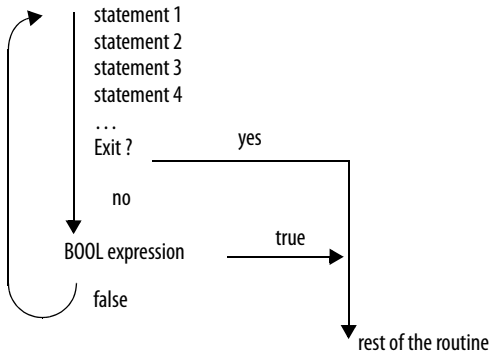
This is the syntax.


REPEAT
<statement>;
statements to execute while bool_expression1 is false
optional
IF bool_expression2 THEN
EXIT;
If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.
END_IF;
UNTIL bool_expression1
END_REPEAT;

These diagrams show how a REPEAT...UNTIL loop executes and how an EXIT statement leaves the loop early.



While the *bool_expression* is false, the controller executes only the statements within the REPEAT...UNTIL loop.



To stop the loop before the conditions are false, use an EXIT statement.

Arithmetic Status Flags

The Arithmetic Status Flags are not affected.

Fault Conditions

A major fault will occur if the construct loops are too long. The fault type is 6 and the fault code is 1.

Example 1

If you want this	Enter this structured text

The REPEAT...UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again.
This differs from the WHILE...DO loop because the WHILE...DO The WHILE...DO loop evaluates its conditions first. If the conditions are true, the controller then executes the statements within the loop. The statements in a REPEAT...UNTIL loop are always executed at least once. The statements in a WHILE...DO loop might never be executed.
	pos := -1;
	REPEAT
	  pos := pos + 2;
	UNTIL ((pos = 101) OR (structarray[pos].value = targetvalue))
	end_repeat;

Example 2

If you want this	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
1. Initialize Element_number to 0.
2. Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
3. Set String_tag[element_number] = the character at SINT_array[element_number].
4. Add 1 to element_number. This lets the controller check the next character in SINT_array.
5. Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.)
6. If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.)
7. If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.
Otherwise, go to step 3.
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	Repeat
	  String_tag.DATA[element_number] := SINT_array[element_number];
	  element_number := element_number + 1;
	  String_tag.LEN := element_number;
	  If element_number = SINT_array_size then
	    exit;
	  end_if;
	Until SINT_array[element_number] = 13
	end_repeat;

Comments

To make your structured text easier to interpret, add comments to it.

- Comments let you use plain language to describe how your structured text works.
- Comments do not affect the execution of the structured text.

Structured text comments are downloaded into controller memory and are available for upload. To add comments to your structured text, follow these steps.

Table 187 - Adding Comments to Structured Text

To Add a Comment	Use One of These Formats
on a single line	//*comment*
at the end of a line of structured text	(**comment*)
	/**comment*/
within a line of structured text	(**comment*)
	/**comment*/
that spans more than one line	(**start of comment . . . end of comment*)
	/**start of comment . . . end of comment*/

These are some examples.

Format	Example

//*comment*	**At the beginning of a line**
	//Check conveyor belt direction
	IF conveyor_direction THEN...
	At the end of a line
	ELSE //If conveyor isn't moving, set alarm light
	light := 1;
	END_IF;

(**comment*)	Sugar.Inlet[:=]1;(*open the inlet*) IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN... (*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*) IF tank.temp > 200 THEN...
/**comment*/	Sugar.Inlet:=0;/*close the inlet*/ IF bar_code=65 /*A*/ THEN... /*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/ SIZE(Inventory,0,Inventory_Items);

Notes:

Motion Instructions and Integrated Motion Control Modes

This table lists the motion instructions and the support for integrated motion related to the Axis Configuration and Feedback Type parameters.

For more information about axis configuration, feedback types, and control modes, see these publications:

- Integrated Motion on the Ethernet/IP Network User Manual, publication MOTION-UM003
- Integrated Motion on the Ethernet/IP Network Reference Manual, publication MOTION-RM003

Symbol	Description
X	Yes
-	No
#	MSO and MDS execution initiate mutually exclusive modes of operation and execution is conditional on mode. Ref detail in MSO and MDS documentation
*	Axis can be used as a master axis reference only for this instruction
C	Axis can conditionally use motion planner instructions if enabled with MSO instruction otherwise would error

Table 188 - Motion Instructions and the Related Control Modes

Motion Instruction		FeedbaCk Only	FrequenCy Control	Position Loop	VeloCity Loop		Torque Loop
			No FeedbaCk		Feedback	No Feedback	
Motion Direct Drive On	MDO	-	-	-	-	-	-
Motion Direct Drive Off	MDF	-	-	-	-	-	-
Motion Servo On	MSO	-	#	X	#	#	#
Motion Servo Off	MSF	-	X	X	X	X	X
Motion Axis Fault Reset	MAFR	X	X	X	X	X	X
Motion Axis Shutdown	MASD	X	X	X	X	X	X
Motion Axis Shutdown Reset	MASR	X	X	X	X	X	X
Motion Drive Start	MDS		#	-	#	#	#
Motion Arm Watch Position	MAW	X	-	X	X	-	X
Motion Disarm Watch Position	MDW	X	-	X	X	-	X
Motion Arm Registration	MAR	X	-	X	X	-	X

Table 188 - Motion Instructions and the Related Control Modes (Continued)

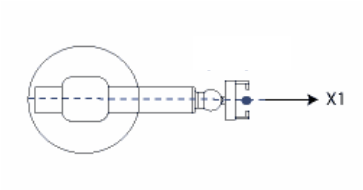
Motion Instruction		FeedbaCk Only	FrequenCy Control	Position Loop	VeloCity Loop		Torque Loop
Motion Disarm Registration	MDR	X	-	X	X	-	X
Motion Arm Output Cam	MAOC	X	-	X	X	-	X
Motion Disarm Output Cam	MDOC	X	-	X	X	-	X
Motion Redefine Position	MRP	X	C	X	C	C	C
Motion Axis Home	MAH	X	-	X	C	-	C
Motion Axis Jog	MAJ	-	C	X	C	C	-
Motion Axis Move	MAM	-	C	X	C	C	-
Motion Change Dynamics	MCD	-	C	X	C	C	-
Motion Axis Stop	MAS	X	X	X	X	X	X
Motion Axis Gear	MAG	*	C	X	C	C	*
Motion Axis Position Cam	MAPC	*	C	X	C	C	*
Motion Axis Time Cam	MATC	-	C	X	C	C	-
Motion Coordinated Linear Move	MCLM	-	C	X	C	C	-
Motion Coordinated Circular Move	MCCM MMM	-	C	X	C	C	-
Motion Coordinated Stop	MCS	X	X	X	X	X	X
Motion Coordinated Shutdown	MCSD	X	X	X	X	X	X
Motion Coordinated Shutdown Reset	MCSR	X	X	X	X	X	X
Motion Coordinated Change Dynamics	MCCD	-	C	X	C	C	-
Motion Coordinated Transform	MCT	-	C	X	C	C	-
Motion Calculate Target Position	MCTP	X	C	X	C	C	X
Motion Master Driven Axis Control	MDAC	*	C	X	C	C	*
Motion Master Driven Coordinated Control	MDCC	*	C	X	C	C	*
Motion Run Axis Tuning	MRAT	-	-	X	X	X	-
Motion Apply Axis Tuning	MAAT	-	-	-	-	-	-
Motion Run Hookup Diagnostic	MRHD	X	X	X	X	X	X
Motion Apply Hookup Diagnostic	MAHD	-	-	-	-	-	-
Motion Group Strobe Position	MGSP	X	X	X	X	X	X
Motion Group Shutdown	MGSD	X	X	X	X	X	X
Motion Group Shutdown Reset	MGSR	X	X	X	X	X	X
Motion Group Stop	MGS	X	X	X	X	X	X

Camming

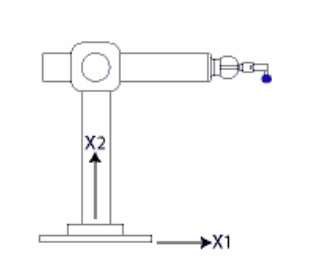
This appendix describes camming concepts. You use the motion coordinated instructions to move up to three axes in a coordinate system. Descriptions of these instructions are located in the Motion Coordinate System User Manual, publication MOTION-UM002.

Topic	Page
Camming Concepts	393
Cam Profiles	395
Scaling Cams	401
Cam Execution Modes	403
Execution Schedule	403
Pending Cams	408

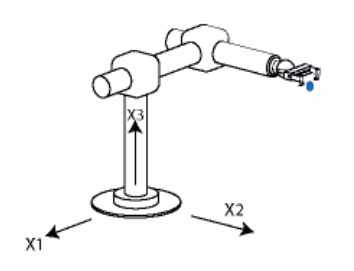
Figure 43 - Coordinate Systems with Orthogonal Axes



Cartesian Coordinate System

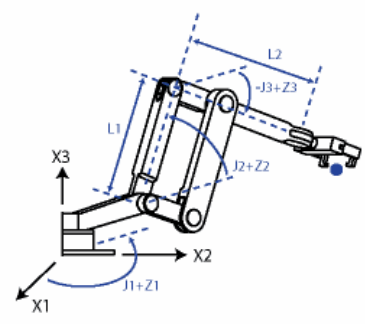


Two-dimensional Cartesian Coordinate System

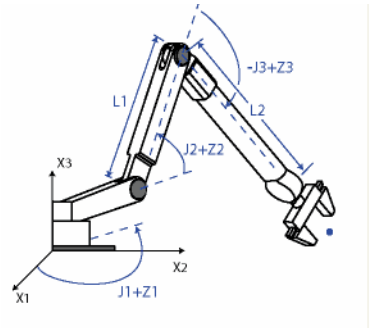


Three-dimensional Cartesian Coordinate System

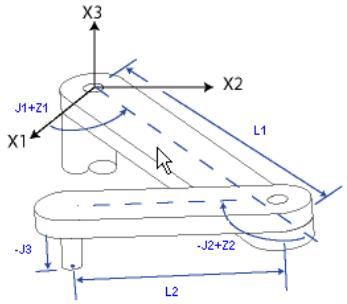
Figure 44 - Coordinate Systems with Nonorthogonal Axes



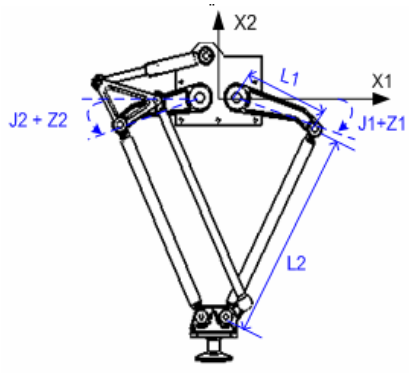
Articulated Dependent Coordinate System



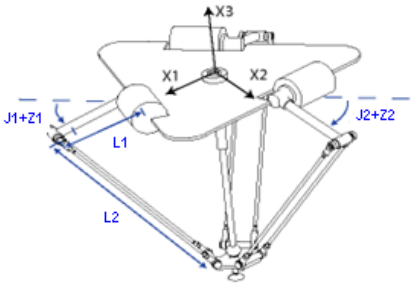
Articulated Independent Coordinate System



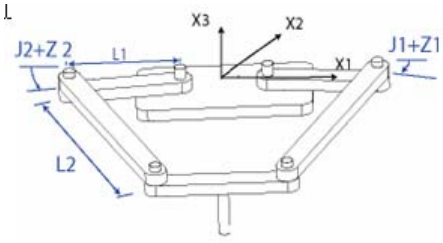
SCARA Independent Coordinate System



Delta Two-dimensional Coordinate System



Delta Three-dimensional Coordinate System



SCARA Delta Coordinate System

Use this table to choose a motion coordinated instruction. Detailed information about these coordinate instructions can be found in the Coordinate System User Manual, publication MOTION-UM002.

If you want to	Use this instruction	Available in these languages
Initiate a single or multi-dimensional linear coordinated move for the specified axes within a Cartesian coordinate system.	Motion Coordinated Linear Move (MCLM)	• Relay ladder • Structured text
Initiate a two- or three-dimensional circular coordinated move for the specified axes within a Cartesian coordinate system.	Motion Coordinated Circular Move (MCCM)	• Relay ladder • Structured text
Initiate a change in path dynamics for coordinate motion active on the specified coordinate system.	Motion Coordinated Change Dynamics (MCCD)	• Relay ladder • Structured text
Stop the axes of a coordinate system or cancel a transform.	Motion Coordinated Stop (MCS)	• Relay ladder • Structured text
Initiate a controlled shutdown of all of the axes of the specified coordinate system.	Motion Coordinated Shutdown (MCSD)	• Relay ladder • Structured text
Start a transform that links two coordinate systems together.	Motion Coordinated Transform (MCT)(1)	• Relay ladder • Structured text
Calculate the position of one coordinate system with respect to another coordinate system.	Motion Calculate Transform Position (MCTP)(1)	• Relay ladder • Structured text
Initiate a reset of all of the axes of the specified coordinate system from the shutdown state to the axis ready state and clear the axis faults.	Motion Coordinated Shutdown Reset (MCSR)	• Relay ladder • Structured text

(1) You cannot use this instruction with SoftLogix controllers.

Camming Concepts

Camming is the process of coordinating the movement of two axes, a master axis and a slave axis, where the movement of one is completely dependent on the movement of the other.

There are two types of camming:

- Mechanical Camming
- Electronic Camming

IMPORTANT Logix5000 motion control applications use electronic camming. A brief description of mechanical camming provides you an understanding of both types of camming. However, the remainder of this publication describes how to use electronic camming in your motion control application.

Mechanical Camming

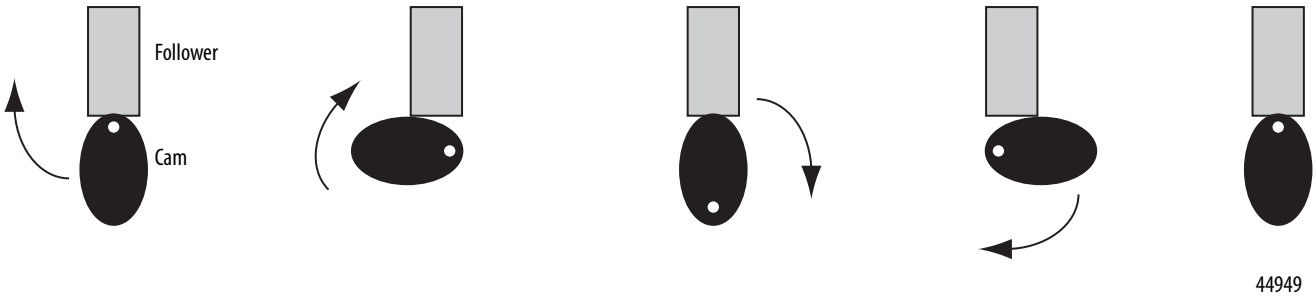
In mechanical camming, the master axis functions as a cam. A cam is an eccentric wheel mounted on a rotating shaft and used to produce variable or reciprocating motion in another engaged part, that is, the slave axis. The slave axis is also known as a follower assembly.

Mechanical camming has the following characteristics:

- There is a physical connection between the cam and the follower.
- The follower conforms to the cam shape as the cam unit rotates.
- Motion is limited by the cam shape.

This figure illustrates a mechanical cam turning in a clockwise manner and the affect it has on a follower that is physically connected to it.

Figure 45 - Mechanical Cams



Electronic Camming

Electronic camming is an electronic replacement for a mechanical camming. In this case, there is still a master axis that produces variable and reciprocating motion in a slave axis. However, electronic camming coordinates the movement of the two separate axes without a physical connection between them. There is no physical cam or follower assembly.

In addition to removing the physical connection between axes, electronic camming:

- Creates coordinated motion profiles that are functions of the time or relative position of another axis.
- Allows you to configure higher cam velocities.
- Is defined by using a 'point pair' table of values. This table is a master axis set of point positioning values and a corresponding set of slave axis point positioning values.

  The user-defined position point array causes one closed-loop axis to move with another open or closed-loop axis.

Cam Profiles

A cam profile is a representation of non-linear motion, that is, a motion profile that includes a start point, end point, and all points and segments in between. A cam profile is represented by an array of cam elements. The point pair used in a cam profile determines slave axis movement in response to master axis positions or times.

In a motion control application, you can use two different types of general cam profiles to accomplish electronic camming:

- Position Cam Profile
- Time Cam Profile

Position Cam Profile

Position-lock cams provide the capability of implementing non-linear electronic gearing relationships between two axes based on a Cam Profile. Upon execution of this instruction, the axis specified as the slave is synchronized with the axis designated as the master.

A position cam profile is defined by using a table of points that contains the following information:

- An array of master axis position values
- An array of slave axis position values

The master axis position values correspond to the slave axis position values. In other words, when the master axis reaches a specific position, the slave axis moves to its specific corresponding point, as defined in the cam profile's table of points.

Additionally, a position cam profile does the following:

- Provides the capability of implementing non-linear electronic gearing relationships between two axes
- Does not use maximum velocity, acceleration, or deceleration limits

Position cam profiles are used with Motion Axis Position Cam (MAPC) instructions. Upon execution of this instruction, the slave axis is synchronized with the master axis.

See Motion Axis Position Cam (MAPC) on page 156 for more information on how to configure the position cam profile in an MAPC instruction.

Linear and Cubic Interpolation

The resultant calculated cam profiles are fully interpolated. This means that if the current master position or time does not correspond exactly with a point in the cam array used to generate the cam profile, the slave axis position is determined by linear or cubic interpolation between adjacent points. In this way, the smoothest possible slave motion is provided. The MCCP instruction accomplishes this by calculating coefficients to a polynomial equation that determines slave position as a function of master position or time.

Each point in the cam array used to generate the position cam profile can be configured for linear or cubic interpolation. Electronic camming remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic camming motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

Time Cam Profile

A time cam profile functions similarly to a cam drum driven by a constant speed motor.

A time cam profile is also defined by using a table of points. However, with the time cam profile, the table contains the following information:

- An array of master axis time values
- An array of slave axis position values

The master axis time values correspond to slave axis position value. When the master axis reaches a specific point in time, the slave axis moves to a specific position as configured in the cam profile.

Time cam profiles are used with Motion Axis Time Cam (MATC) instructions. Upon execution of this instruction, the slave axis is synchronized with the master axis.

See Motion Axis Time Cam (MATC) on page 178 for more information on how to configure the position cam profile in an MATC instruction.

Linear and Cubic Interpolation

Time cams are fully interpolated. This means that if the current master time value does not correspond exactly with a point in the cam table associated with the cam profile, the slave axis position is determined by linear or cubic interpolation between the adjacent points. In this way, the smoothest possible slave motion is provided.

Each point in the cam array that was used to generate the time cam profile can be configured for linear or cubic interpolation.

Electronic camming remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic camming motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

Calculating a Cam Profile

You can use a Motion Calculate Cam Profile (MCCP) instruction to calculate a cam profile based on an array of cam points. You can establish an array of cam points programmatically or by using the Logix Designer software Cam Profile Editor. Each cam point in the cam array consists of a slave position value, a master position (position cam) or time (time cam) value, and an interpolation type (linear or cubic). An MAPC or MATC instruction can use the resulting cam profile to govern the motion of a slave axis according to master position or time.

Using Common Cam Profiles

There are four common cam profiles that can be used as position cam or time cam profiles:

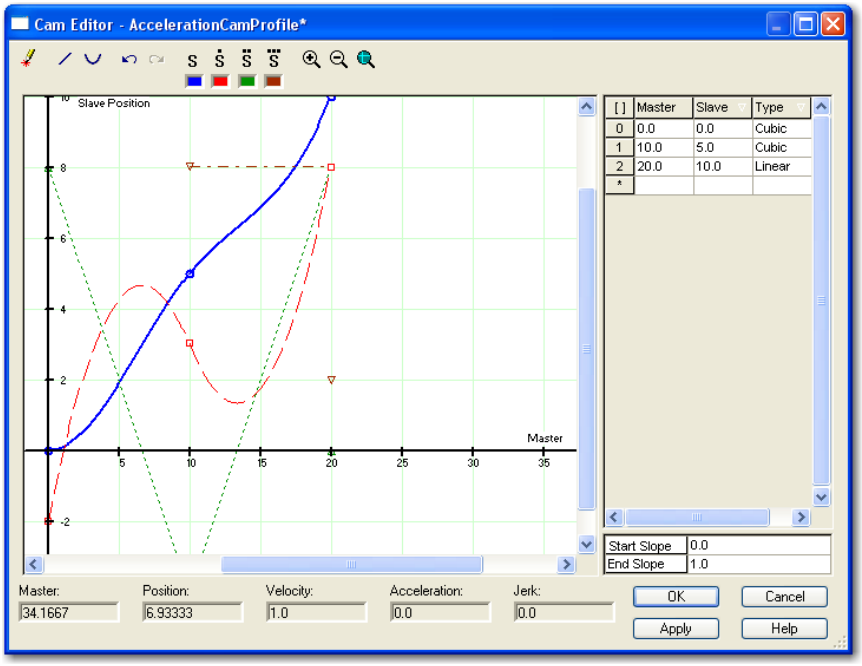
- Acceleration Cam Profile
- Run Cam Profile
- Deceleration Cam Profile
- Dwell Cam Profile

Cam profiles are configured for each required slave axis change of position, as corresponds to specific master axis position or time positions.

Acceleration Cam Profile

An acceleration cam profile determines a slave axis acceleration to a particular position. This graphic illustrates a sample acceleration cam profile in the Logix Designer programming software cam editor.

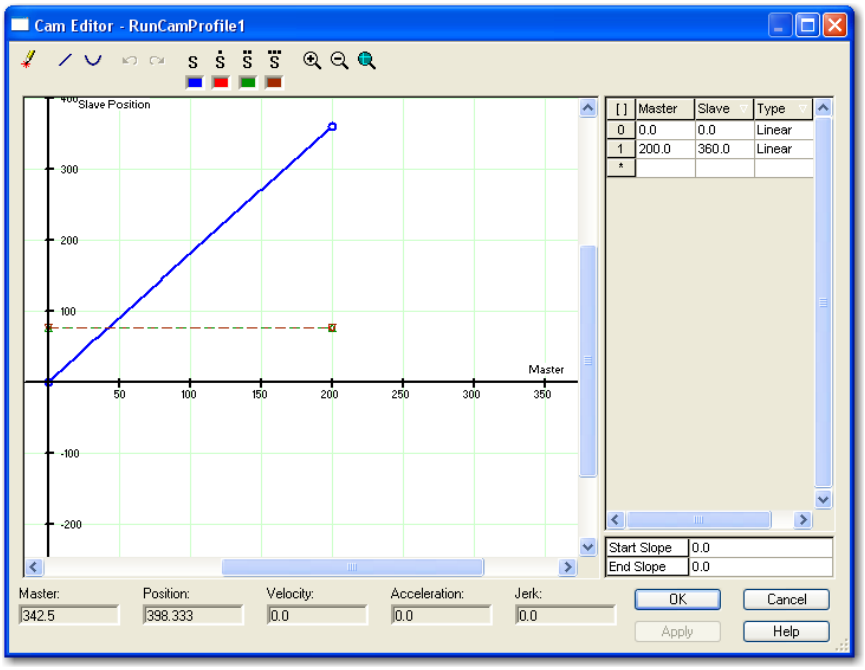
Figure 46 - Acceleration Cam Profile



Run Cam Profile

A run cam profile determines a slave axis' movement that begins when the master axis reaches a specific position and remains steady until the end of the cam profile. This graphic illustrates a sample run cam profile in the Logix Designer programming software cam editor.

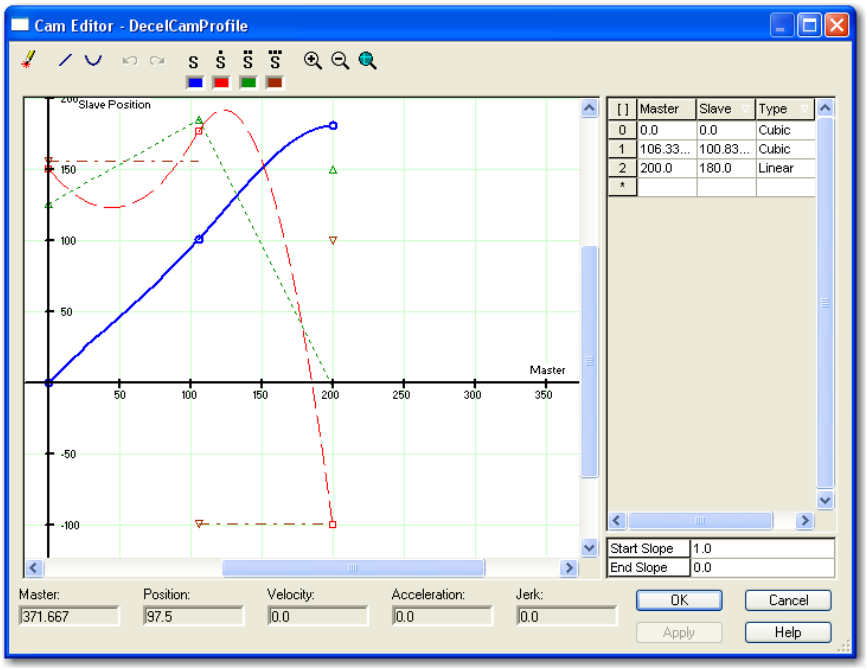
Figure 47 - Run Cam Profile



Deceleration Cam Profile

A deceleration cam profile determines a slave axis' deceleration from a particular position. This graphic illustrates a sample deceleration cam profile in the Logix Designer programming software cam editor.

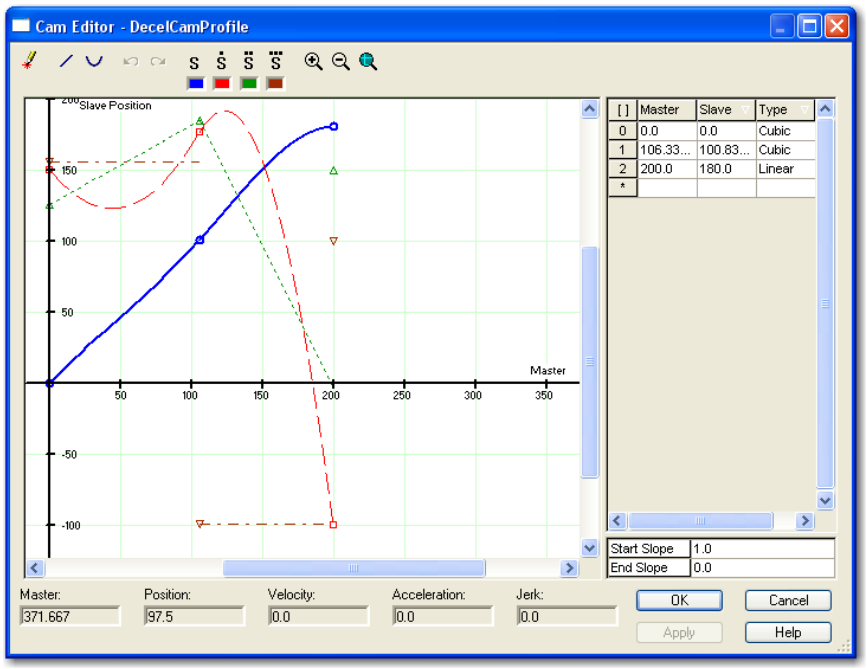
Figure 48 - Deceleration Cam Profile



Dwell Cam Profile

A dwell cam profile stops all slave axis movement until another cam profile begins operation. Typically, a dwell cam profile follows a deceleration cam profile. This graphic illustrates a sample dwell cam profile in the Logix Designer programming software cam editor.

Dwell Cam Profile



Behavior of Pending Cams

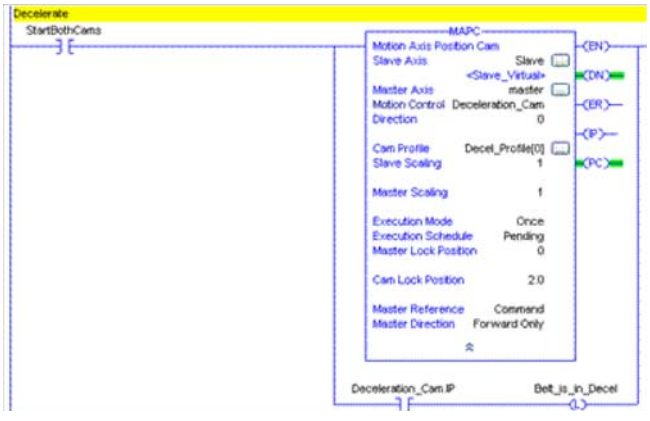
If you want to run one profile and then pend another one, you need to execute the MAPC instructions in the right order.

For example, if you want to run only one slave cycle, start with the Accel_Profile and pend the Decel_Profile immediately, that results in 2 x ½ Cycle = 1 Cycle. Both of these are executed at the same point in time:

- Set the execution schedule in the MAPC instruction for Acceleration as Immediate.
- Set the Deceleration to Pending.

Execution Schedule: Immediate

Execution Schedule: Pending



Scaling Cams

You can use the scaling feature to determine the general form of the motion profile with a single stored cam profile. With this feature, one standard cam profile can be used to generate a family of specific cam profiles.

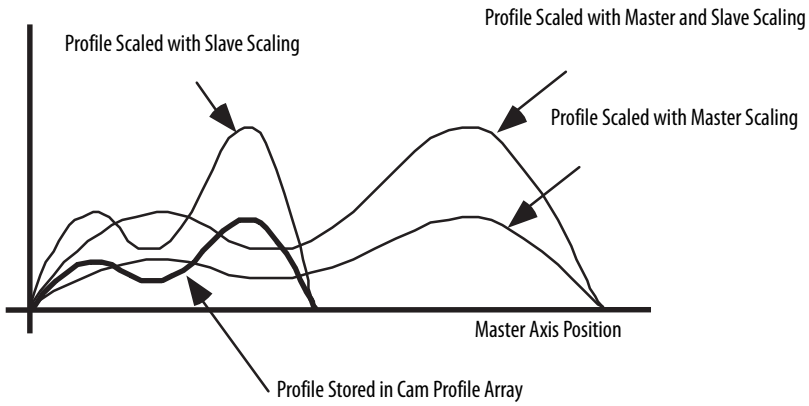
Scaling works slightly differently when it is used with an MAPC instruction, that is, in position cam profiles, than when it is used with an MATC instruction, that is, in time cam profiles.

Scaling Position Cam Profiles

A position cam profile can be scaled in both the master dimension and slave dimension when it is executed. The scaling parameters are then used to define the total master or slave travel over which the profile is executed, as shown in Figure 49.

When an MAPC instruction specifies a position cam profile array, the master and slave values defined by the cam profile array take on the position units of the master and slave axes respectively. By contrast, the Master and Slave Scaling parameters are 'unit-less' values that are simply used as multipliers to the cam profile.

Figure 49 - Cam Profile Array


Profile Scaled with Master and Slave Scaling
Profile Scaled with Slave Scaling
Profile Scaled with Master Scaling
Master Axis Position
Profile Stored in Cam Profile Array

By default, both the Master Scaling and Slave Scaling parameters are set to 1. To scale a position cam profile, enter a Master Scaling or Slave Scaling value other than 1.

Increasing the Master Scaling value of a position cam profile **decreases** the velocities and accelerations of the profile. However, increasing the slave scaling value **increases** the velocities and accelerations of the profile.

To maintain the velocities and accelerations of the scaled profile approximately equal to those of the unscaled profile, the Master Scaling and Slave Scaling values should be equal. For example, if the Slave Scaling value of a profile is 2, the Master Scaling value should also be 2 to maintain approximately equal velocities and accelerations during execution of the scaled position cam.

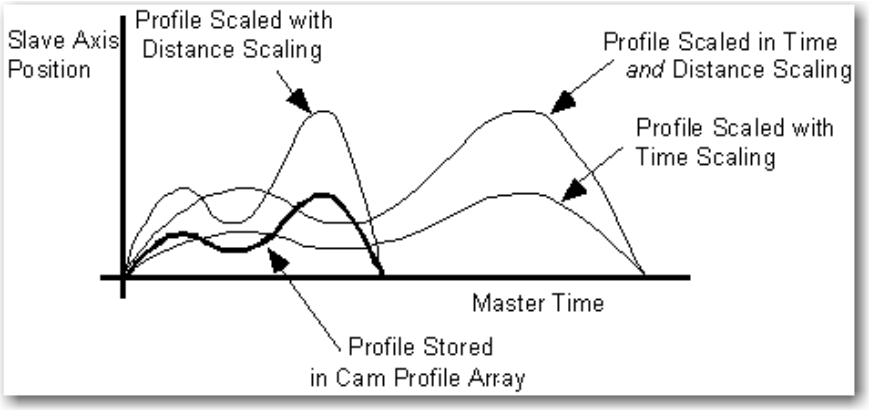
ATTENTION: Decreasing the Master Scaling value or increasing the Slave Scaling value of a position cam increases the required velocities and accelerations of the profile. This can cause a motion fault if the capabilities of the drive system are exceeded.

Scaling Time Cam Profiles

A time cam profile can be scaled in both time and distance when it is executed.

When an MATC instruction specifies a time cam profile array, the master coordinate values defined by the cam profile array take on the time units and the slave values take on the units of the slave axis. By contrast, the Time and Distance Scaling parameters are 'unitless' values that are simply used as multipliers to the cam profile.

Figure 50 - Scaling Time Cam Profile


Slave Axis Position
Profile Scaled with Distance Scaling
Profile Scaled in Time *and* Distance Scaling
Profile Scaled with Time Scaling
Master Time
Profile Stored in Cam Profile Array

By default, both the Time and Distance Scaling parameters are set to 1. To scale a time cam profile, enter a Time Scaling or Distance Scaling value other than 1.

Increasing the Time Scaling value of a time cam profile **decreases** the velocities and accelerations of the profile. However, increasing the Distance Scaling value **increases** the velocities and accelerations of the profile.

To maintain the velocities and accelerations of the scaled profile approximately equal to those of the unscaled profile, the Time Scaling and Distance Scaling values should be equal. For example, if the Distance Scaling value of a profile is 2, the Time Scaling value should also be 2 to maintain approximately equal velocities and accelerations during execution of the scaled time cam.

IMPORTANT Decreasing the Time Scaling value or increasing the Distance Scaling of a time cam increases the required velocities and accelerations of the profile. This can cause a motion fault if the capabilities of the drive system are exceeded.

Cam Execution Modes

Cam execution modes determine if the cam profile is executed only one time or repeatedly. You must configure the Execution Mode parameter on an MAPC or MATC instruction.

Table 189 - Execution Mode Descriptions

Execution Mode	Description
Once	Cam motion of slave axis starts only when the master axis moves into the range defined by the start and end points of the cam profile. When the master axis moves beyond the defined range, cam motion on the slave axis stops and the Process Complete bit is set. Slave motion does not resume if the master axis moves back into the cam profile range.
Continuous	Once started the cam profile is executed indefinitely. In this mode, the master and slave positions are unwound when the position of the master axis moves outside the profile range. This unwinding causes the cam profile to repeat. This feature is useful in rotary applications where it is necessary that the cam position run continuously in a rotary or reciprocating fashion.
Persistent(1)	The cam motion of the slave axis proceeds only when the master axis moves within the range defined by the start and end points of the cam profile. When the master axis moves beyond the range of the profile, cam motion on the slave axis stops and only resumes when the master moves back into the profile range specified by the start and end points.

(1) This selection is only available on the MAPC instruction.

Execution Schedule

The Execution Schedule parameter controls an instruction's execution. You must configure the Execution Schedule parameter on an MAPC or MATC instruction. The Execution Schedule selections are different depending on which instruction, that is, the MAPC instruction or the MATC instruction, you are using.

MAPC Instruction

The Execution Schedule parameter selections are the following:

- Immediate
- Pending
- Forward Only
- Reverse Only

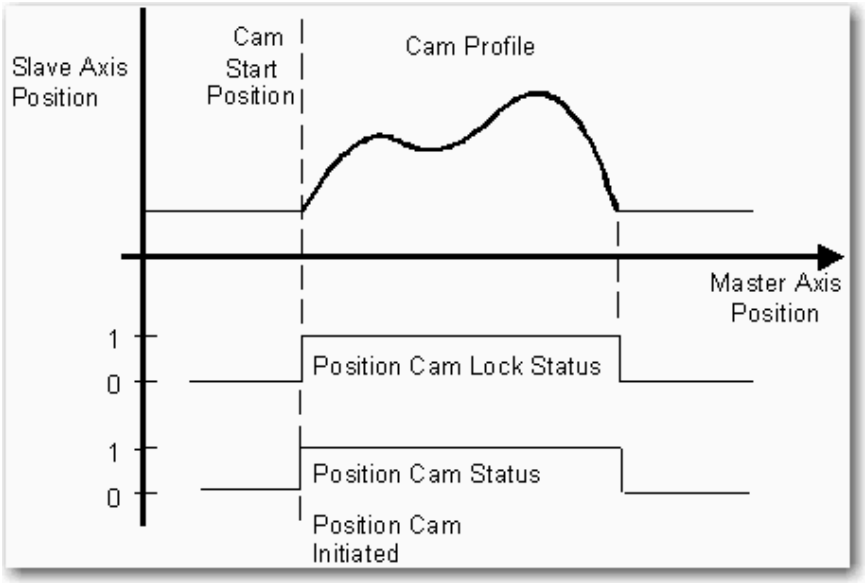
- Bidirectional

Immediate

By default, the MAPC instruction is scheduled to execute Immediately. In this case, there is no delay to the enabling of the position camming process and the Master Lock Position parameter is irrelevant. The slave axis is immediately locked to the master axis beginning at the Cam Lock Position of the specific cam profile.

When the MAPC instruction is executed, the camming process is initiated on the specified slave axis and the Position Cam Status bit in the slave axis' Motion Status word is set. If the Execution Schedule is Immediate, the slave axis is immediately locked to the master according to the specified Cam Profile. This is indicated by the fact that the Position Cam Lock Status bit for the specified slave axis is also set.

Figure 51 - Immediate Execution

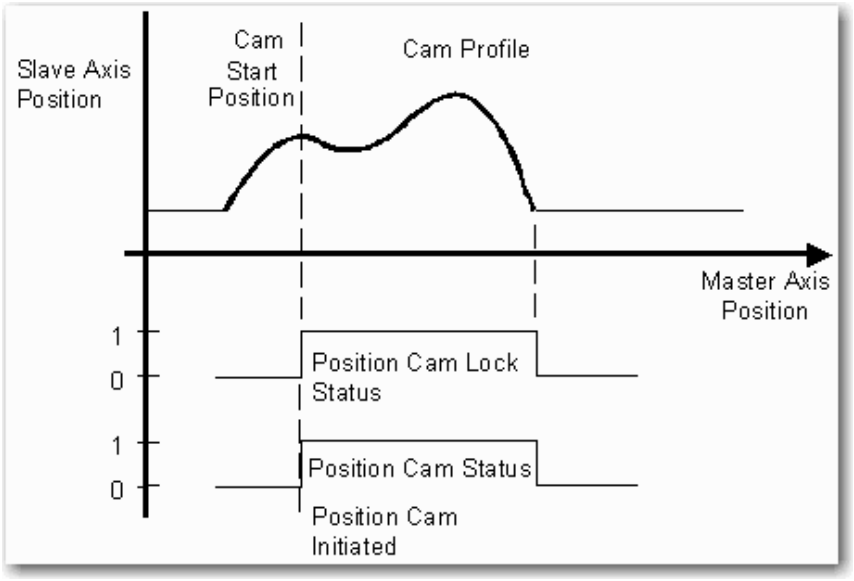


Changing the Cam Lock Position on an MAPC Immediate Execution Schedule

The Cam Lock Position parameter of the MAPC instruction determines the starting location within the cam profile when the slave locks to the master. Typically, the Cam Lock Position is set to the beginning of the cam profile. Because the starting point of most cam tables is 0, the Cam Lock Position is typically set to 0. Alternatively, the Cam Lock Position can be set to any position within the master range of the cam profile. If a Cam Lock Position is specified that is out of this range, the MAPC instruction errors.

The diagram Changing the Cam Lock Position shows the effect of specifying a Cam Lock Position value other than the starting point of the cam table; in this case, the value represents a position within the cam profile itself. Be careful not to define a Cam Start Point that results in a velocity or acceleration discontinuity to the slave axis if the master axis is currently moving.

Figure 52 - Changing the Cam Lock Position



Pending

An MAPC instruction's execution can be deferred pending completion of a currently executing position cam. You can use Execution Schedule selection of Pending to blend two position cam profiles together without stopping motion.

This Execution Schedule selection of Pending is fully described in Pending Cams on page 408.

Forward Only, Reverse Only, or Bidirectional Execution Schedules

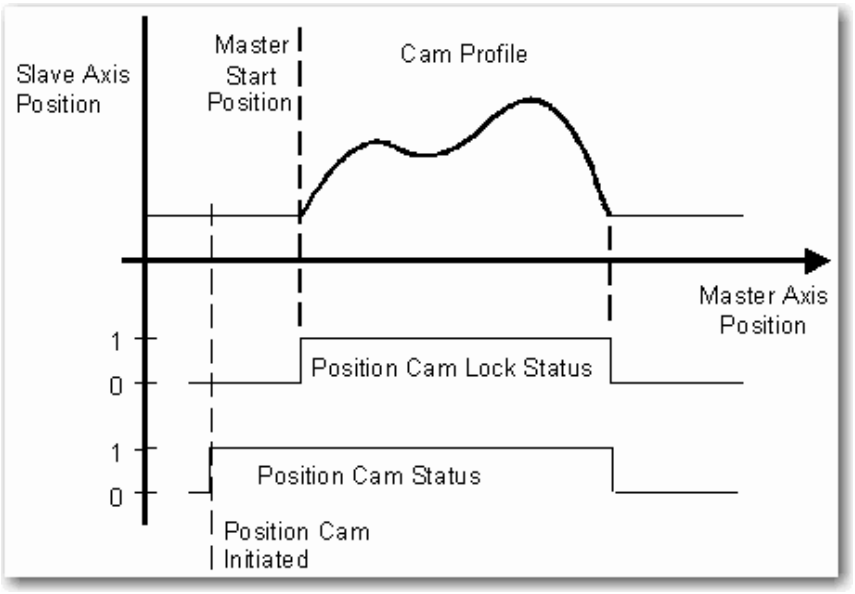
The slave axis is not locked to the master until the master axis satisfies the condition specified when the Execution Schedule parameter is set to any of the following parameters:

- Forward Only
- Reverse Only
- Bidirectional

With any of these selections, the master axis is monitored by the camming process to determine when the master axis passes the specified Master Lock Position in the specified direction. In a rotary axis configuration, this lock criterion is still valid, independent of the turns count.

IMPORTANT If the position reference of the master axis is redefined, for example, with a Motion Redefine Position (MRP) instruction, after the MAPC instruction executes but before the lock condition is satisfied, the cam profile generator monitors the master axis based on the absolute position reference system in effect prior to the redefine position operation.

Figure 53 - Forward Only, Reverse Only, or Bidirectional Execution



When the absolute position of the master axis passes the specified Master Lock Position in the specified direction, the Position Cam Status bit of the Motion Status word for specified slave axis is set. Slave axis motion is then initiated according to the specified cam profile starting at the specified Cam Lock Position of the cam profile.

From this point on, only the **incremental change** in the master axis position determines the corresponding slave axis position from the defined cam profile. This is important for applications where the master axis is a rotary axis because the position cam is then unaffected by the position unwind process.

When the master axis moves out of the range defined by the cam profile, that is, assuming Execution Mode is Once, both the Position Cam Lock Status and the Position Cam Status bits of the Motion Status word are cleared. This Motion Status bit condition indicates that the cam process has completed. This fact is also reflected in the bit leg behavior of the associated MAPC instruction, PC bit set and IP bit clear.

After position cam motion is started when the master axis passes the specified Master Lock Position in either the Forward Only or Reverse Only direction, the master axis can change direction and the slave axis reverses accordingly.

If an MAPC instruction is executed on a slave axis that is actively position camming, an Illegal Dynamic Change error is generated (error code 23). However, this error does not occur if the Execution Schedule is Pending.

MATC Instruction

An MATC instruction uses one of two Execution Schedule parameters:

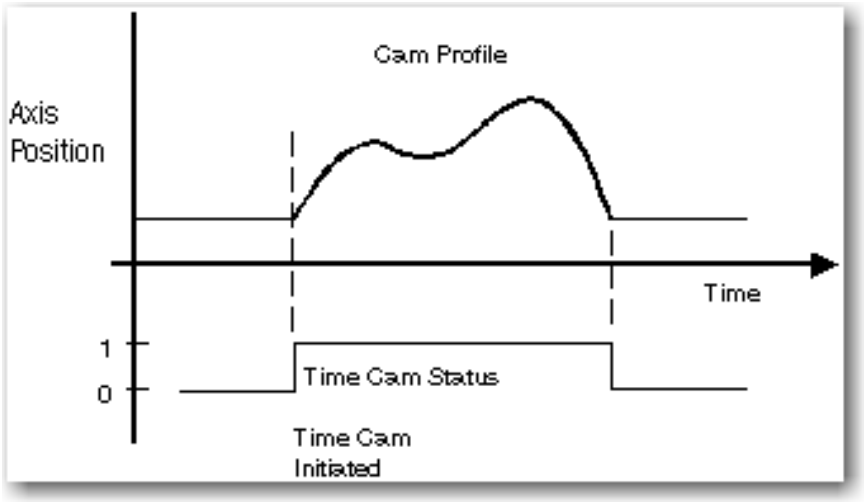
- Immediate
- Pending

Immediate

By default, the MATC instruction is scheduled to execute immediately by virtue of the fact that the default setting of Execution Schedule is Immediate. In this case, there is no delay to the enabling of the time camming process.

As illustrated in Figure 54, when the MATC instruction is executed, the camming process is initiated on the specified axis and the Time Cam Status bit in the axis' Motion Status word is set. If the Execution Schedule parameter is set to Immediate, the axis is immediately locked to the time master coordinate according to the specified Cam Profile.

Figure 54 - Time Cam Status



If an MATC instruction is executed on an axis that is already actively time camming, an Illegal Dynamic Change error is generated (error code 23). The only exception for this is if the Execution Schedule is specified as pending.

Pending

An MATC instruction's execution can be deferred pending completion of a currently executing time cam profile. You can use Execution Schedule selection of Pending to seamlessly blend two time cam profiles together without stopping motion.

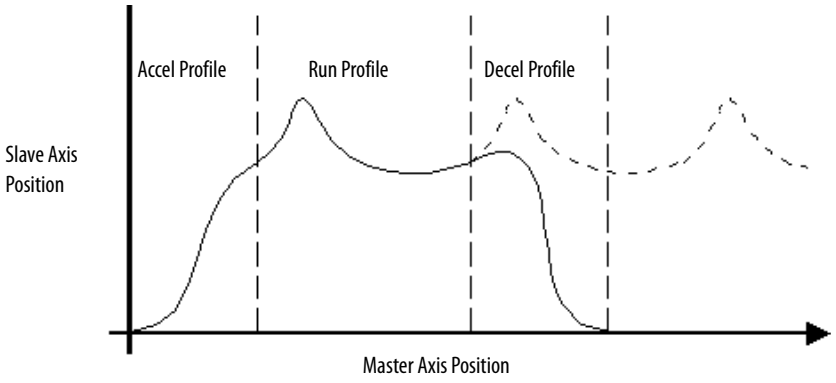
Pending Cams

Cam pending is a technique that lets the blending of one cam profile together with another without stopping either master or slave axis movement. An Execution Schedule selection of Pending can thus be used to seamlessly blend two position cam profiles together without stopping motion.

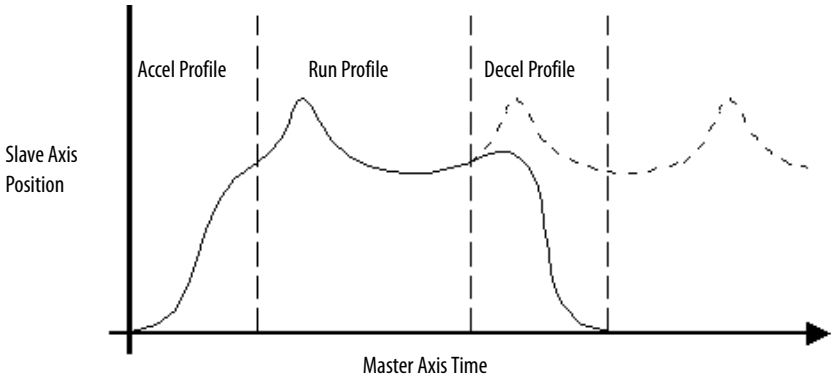
The Pending execution feature is particularly useful when the axis must be accelerated up to speed by using a specific velocity profile. When this acceleration profile is done, it must be smoothly blended into the operating cam profile, that is typically executed continuously. To stop the slave axis, the operating cam profile is smoothly blended into a deceleration profile such that the axis stops at a known location, as shown in graphic Pending Cam Execution.

Figure 55 - Pending Cam Execution

MAPC Instruction



MATC Instruction



By executing the position cam profile as a Pending cam profile while the current profile is still executing, the appropriate cam profile parameters are set up ahead of time. This makes the transition from the current profile to the pending profile seamless; synchronization between the master and slave axes is maintained. To ensure smooth motion across the transition, however, the profiles must be designed such that no position, velocity, or acceleration discontinuities exist between the end of the current profile and the start of the new one. This is done by using the Logix Designer Cam Profile Editor.

Once a pending position cam instruction has been executed, the new cam profile takes effect automatically (and becomes the current profile) when the master axis passes through either the start or end point of the current profile. If the current cam is configured to execute once, the new profile is initiated at the completion of the current cam profile and the PC bit of the currently active instruction (either MAPC or MATC) is set.

If the current cam is configured to execute continuously, the new profile is initiated at the completion of the current pass through the current cam profile and the IP bit of the currently active instruction is cleared. The motion controller keeps track of the master axis position or time, depending on which instruction is used, and the slave axis position relative to the first profile at the time of the

change and uses this information to maintain synchronization between the profiles.

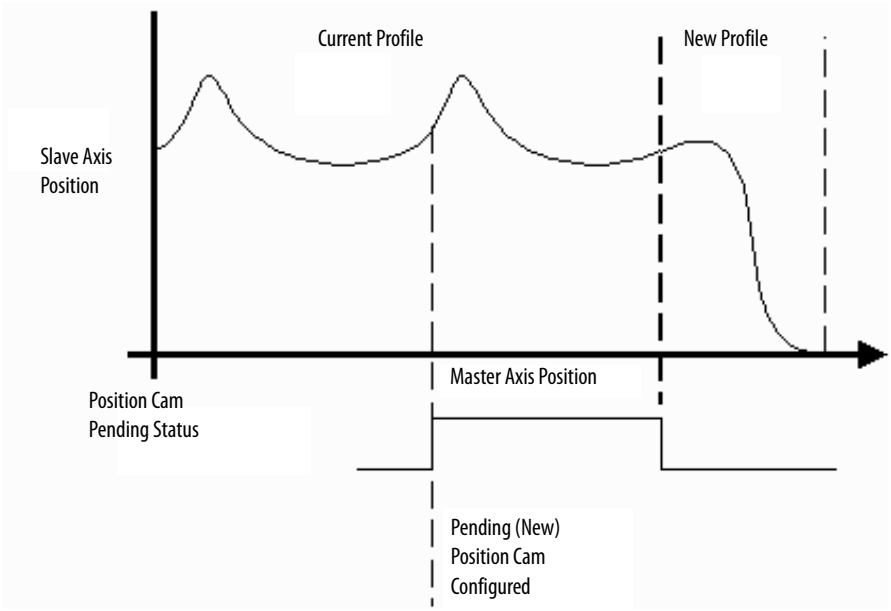
If the Execution Schedule of an instruction is set to Immediate and a position or time cam profile is currently in process, the instruction errs. In this case, the instruction generates an Illegal Dynamic Change error, that is error code 23, in Logix Designer programming software. This error even occurs when the axis is waiting to lock onto the master axis.

If an Execution Schedule of Pending is selected without a corresponding position or time cam profile in progress, the instruction executes but no camming motion occurs until another instruction with a non-pending Execution Schedule is initiated. This allows pending cam profiles to be preloaded prior to executing the initial cam. This method addresses cases where immediate cams would finish before the pending cam could be reliably loaded.

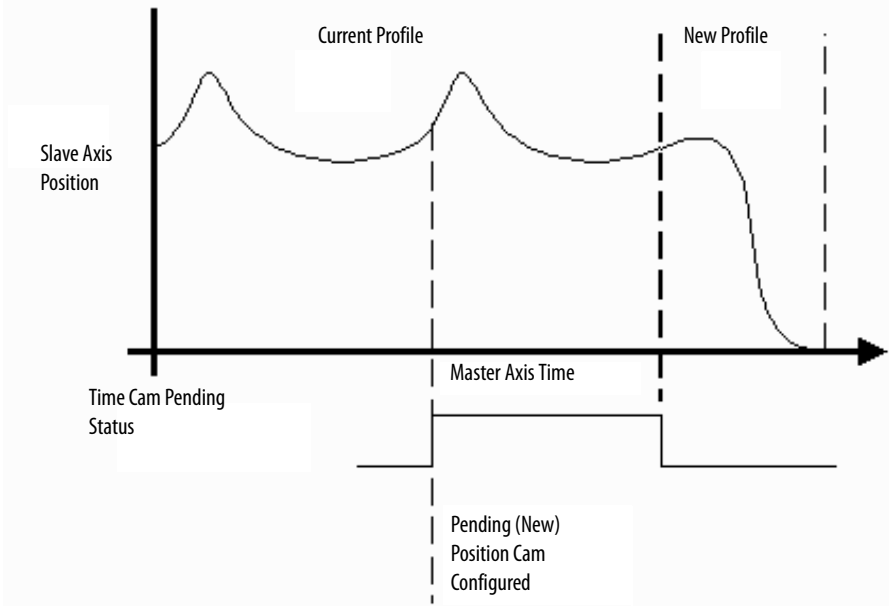
After a Pending position cam has been configured, the Position or Time Cam Pending Status bit of the Motion Status word for the specified slave axis is set to 1 (true). When the pending (new) profile is initiated and becomes the current profile, Position or Time Cam Pending Status bit is immediately cleared as shown in Figure 56.

Figure 56 - Pending Position Cam

MAPC Instruction



MATC Instruction



Notes:

History of Changes

This appendix summarizes the revisions to this manual. Reference this appendix if you need information to determine what changes have been made across multiple revisions. This may be especially useful if you are deciding to upgrade your hardware or software based on information added with previous revisions of this manual.

MOTION-RM002D-EN-P

These are the major changes made in MOTION-RM002D.

Table 190 - Summary of Changes for MOTION-RM002D

Topic
Corrected link to disable gearing information
Updated MAPC Motion_Instruction Structure Descriptions
Added link for the Position-based Output Control with the MAOC Instruction Application Technique, publication 1756-AT017
Corrected equation in Mode Effects on an Output Bit Mode
Added Tuning Configuration Bits
Updated support information
Updated text and formatting
Corrected link to disable gearing information

MOTION-RM002C-EN-P

These are the major changes made in MOTION-RM002C.

Table 191 - Summary of Changes for MOTION-RM002C

Topic
Motion Servo Off (MSF)
Motion Drive Start (MDS)
MAH Extended Error Codes
MAG Extended Error Codes
MCD Extended Error Codes
MAPC Extended Error Codes
Motion Arm Watch (MAW) Description
Motion Arm Registration (MAR) Description
Motion Arm Output Cam (MAOC)

Table 191 - Summary of Changes for MOTION-RM002C

Topic
Motion Disarm Output Cam (MDOC)
Motion Apply Hookup Diagnostics (MAHD)
Motion Run Hookup Diagnostics (MRHD)
MDSC Functionality
Master Drive Axis Control (MDAC)
Master Driven and Time Driven Modes
Error Codes (ERR) for Motion Instructions
MOTION_GROUP Structure
Motion Instructions and Integrated Motion Control Modes
Camming

A

acceleration 318, 397
acceleration cam 397
axis control 303

B

behavior 400
bidirectional 403

C

calulating 397
cam 404
  master axis 395
CAM data type 353
cam execution modes 403
cam pending 408
cam profile 395, 397
cam profile array 401
cam profile editor 397
cam profiles
  scaling time 402
cam shape 394
CAM_PROFILE data type 353
cams 401
changing 404
CIP Sync
  33
coefficients
  calculating 396
common cam profiles 397
control words 303
cubic 396

D

data types 353-358
  CAM 353
  CAM_PROFILE 353
  MOTION_GROUP 354
  MOTION_INSTRUCTION 356
  OUTPUT_CAM 357
  OUTPUT_COMPENSATION 358
deceleration 318, 397
deceleration cam 399
dimension 401
Direct Commands
  supported commands 31
dwell 324, 397
dwell cam 399

E

electronic 393
electronic camming 394
error 407, 408, 410
error code 23 408, 410
error codes 345-352
errors
  conflict 305
  home in progress 305
  inhibited 305
  no master axis 305
  not configured 305
  redefine 305
  same axis 305
  time based programming 324
execution 409
execution schedule parameter 403
execution schedule parameters 407

F

forward only 403

I

illegal dynamic change 407, 408, 410
immediate 403, 407
immediate execution 404
immediate motion instructions 26
interpolated 396
interpolation 396
  between adjacient points 396

J

jerk 319
  troubleshoot 331-343
  tune 327-329
  units 320
jog
  execution of 396

K

kinematics
  See multi-axis coordinated motion instructions

L

linear 396
lock position 404
Logix Designer Cam Profile Editor 409

M

MAPC 395, 400, 401, 403, 404, 405, 407, 409, 411
  cam profile 397

master 298, 404
changing 306
relationship 302
stopping 308
master axis 394, 404, 411
execution mode 403
position values 395
time cam profile 396
master axis position 409
master axis time 409
Master Driven Axis Control 297
Master Driven Speed Control 297
MATC 401, 402, 403, 407, 409, 411
cam profile 397
time cam 396
MCCP 396, 397
MDAC 297, 302, 303, 306
Master
Slave relationship 297
MDAC1 298
MDS 74
MDSC 297
parameters 309
calculated data 309, 313
event distance 309
instruction mode 309
lock direction 309, 310
lock position 309, 311
program rate 316
program time 317
synchronization of axes 297
Time Driven CAM profiles 297
mechanical 393
mechanical camming 393
message type instructions 28
mode
master driven 305
time driven 305
motion
immediate type instructions 26
message type instructions 28
process type instructions 29
Motion Apply Axis Tuning (MAAT) 32, 261-267
arithmetic status flags 265
changes to status bits 267
description 263
error codes 267
execution conditions 266
fault conditions 265
operands 262
Motion Apply Hookup Diagnostics (MAHD) 32, 278-283
arithmetic status flags 281
changes to status bits 283
description 280
Encoder Hookup test 281
error codes 283
execution conditions 282
fault conditions 281
Motor Encoder Hookup test 280
operands 278
Motion Arm Output Cam (MAOC) 31, 239-255
arithmetic status flags 251
axis and module fault conditions disarm output cams 254
axis arm and cam arm positions 244
changes to status bits 254
description 242
error codes 252
execution conditions 252
fault conditions 251
latch type 245
mode compensation 250
offset and delay compensation 249
operands 239
output cam array checks 248
output compensation array checks 251
specifying output compensation 248
unlatch type 246
Motion Arm Registration (MAR) 31, 227-235
arithmetic status flags 231
changes to status bits 235
description 229
error codes 234
execution conditions 233
fault conditions 232
rearming an MAR instruction 230
Motion Arm Watch (MAW) 31, 216-222
arithmetic status flags 219
changes to status bits 222
description 218
error codes 222
execution conditions 220
fault conditions 219
operands 216
Motion Axis Fault Reset (MAFR) 31, 75-79
arithmetic status flags 77
changes to status bits 78
description 76
error codes 78
example 78
execution conditions 77
fault conditions 77
operands 75
Motion Axis Gear (MAG) 31, 123-132
arithmetic status flags 129
changes to status bits 132
description 126
error codes 131
example 132
execution condition 130
extended error codes 131
fault conditions 129
operands 123
programming guidelines 127
Motion Axis Home (MAH) 31, 92-98
arithmetic status flags 95
changes to status bits 98
description 93
error codes 97
execution conditions 96
fault conditions 95
operands 92

Motion Axis Jog (MAJ) 31
arithmetic status flags 105
changes to status bits 106
description 103
error codes 105
fault conditions 105
operands 99
Motion Axis Move (MAM) 31, 111-121
arithmetic status flags 119
changes to status bits 120
description 114
error codes 120
example 121
fault conditions 119
operands 111
programming guidelines 114
Motion Axis Position Cam 395
Motion Axis Position Cam (MAPC) 31, 156-177
arithmetic status flags 173
changes to status bits 176
changing the cam lock position 165, 404
description 160
error codes 175
example 177
execution conditions 174
execution schedule 163
extended error codes 175
fault conditions 173
fault recovery 172
forward only, reverse only, or bidirectional execution 166, 405
immediate execution 163, 404
incremental moves 171
linear and cubic interpolation 161
master direction 170
master offset moves 171
master reference 169
merging from a cam 172
moving while camming 171
operands 156
pending cam execution 167
scaling position cams 162
specifying the cam profile 161
stopping a cam 171, 187
Motion Axis Shutdown (MASD) 31, 48-53
arithmetic status flags 50
changes to status bits 52
description 49
error codes 52
example 52
execution conditions 51
fault conditions 50
operands 48
Motion Axis Shutdown Reset (MASR) 31, 54-58
arithmetic status flags 56
changes to status bits 57
description 55
error codes 57
example 58
execution conditions 56
fault conditions 56
operands 54
Motion Axis Stop (MAS) 31, 82-88
arithmetic status flags 87
changes to status bits 88
description 85
error codes 87
fault conditions 87
operands 82
Motion Axis Time Cam (MATC) 31, 178-191
arithmetic status flags 188
cam profile array checks 182
cam profile execution modes 184
changes to status bits 191
description 181
error codes 190
example 191
execution conditions 189
execution schedule 184
extended error codes 190
fault conditions 188
immediate execution 184
linear and cubic interpolation 181
merging from a cam 188
operands 178
pending cam execution 185
scaling time cams 183
specifying the cam profile 182
Motion Calculate Cam Profile (MCCP) 31, 150-155
arithmetic status flags 154
calculating the cam profile 153
changes to status bits 154
description 151
error codes 154
example 155
extended error codes 154
fault conditions 154
linear and cubic spline interpolation 152
operands 150
specifying the cam profile tag 152
start slope and end slope 152
Motion Calculate Slave Values (MCSV) 31, 192-194
arithmetic status flags 193
changes to status bits 194
description 193
error codes 194
extended error codes 194
fault conditions 193
operands 192
Motion Change Dynamics (MCD) 31, 133-141
absolute mode 144
actual position 145
arithmetic status flags 139
changes to status bits 140
command position 145
description 136
error codes 140
example 141, 149
execution conditions 139
extended error codes 140, 148
fault conditions 139
operands 133
programming guidelines 144
relative mode 145

Motion Configuration Instructions 261-296
Motion Apply Axis Tuning (MAAT) 32, 261-267
Motion Apply Hookup Diagnostics (MAHD) 32, 278-283
Motion Run Axis Tuning (MRAT) 32, 268-277
Motion Run Hookup Diagnostics (MRHD) 32, 284-296
Motion Coordinated Instructions
Motion Coordinated Change Dynamics (MCCD) 32, 393
Motion Coordinated Circular Move (MCCM) 32, 393
Motion Coordinated Linear Move (MCLM) 32, 393
Motion Coordinated Shutdown (MCSD) 32, 393
Motion Coordinated Stop (MCS) 32, 393
motion coordinated instructions
See multi-axis coordinated motion instructoins
Motion Direct Drive Off (MDF) 31, 65-68
arithmetic status flags 66
changes to status bits 68
description 66, 70
error codes 68
example 68
execution conditions 67
fault conditions 66
operands 65, 69
Motion Direct Drive Off (MDS)
example 74
Motion Direct Drive On (MDO) 31, 59-64
arithmetic status flags 62
changes to status bits 64
description 60
error codes 64
example 64
execution conditions 63
extended error codes 64
fault conditions 62
loss of feedback 62
operands 59
Motion Disarm Output Cam (MDOC) 31, 256-260
arithmetic status flags 257
changes to status bits 259
description 257
error codes 258
execution conditions 258
fault conditions 257
operands 256
Motion Disarm Registration (MDR) 31, 236-238
arithmetic status flags 237
changes to status bits 238
description 237
error codes 237
fault conditions 237
operands 236
Motion Disarm Watch (MDW) 31, 223-226
arithmetic status flags 224
description 224
execution conditions 225
fault conditions 224
operands 223
Motion Drive Start (MDS) 31, 69-74
arithmetic status flags 73
changes to status bits 74
error codes 74
fault conditions 74
Motion Event Instructions 215-260
Motion Arm Output Cam (MAOC) 31, 239-255
Motion Arm Registration (MAR) 31, 227-235
Motion Arm Watch (MAW) 31, 216-222
Motion Disarm Output Cam (MDOC) 31, 256-260
Motion Disarm Registration (MDR) 31, 236-238
Motion Disarm Watch (MDW) 31, 223-226
Motion Group Instructions 195
Motion Group Shutdown (MGSD) 31, 202-205
Motion Group Shutdown Reset (MGSR) 31, 206-209
Motion Group Stop (MGS) 31, 196-201
Motion Group Strobe Position (MGSP) 31, 210-213
Motion Group Shutdown (MGSD) 31, 202-205
arithmetic status flags 203
changes to status bits 205
description 203
error codes 205
example 205
execution conditions 204
fault conditions 204
operands 202
Motion Group Shutdown Reset (MGSR) 31, 206-209
arithmetic status flags 207
changes to status bits 209
description 207
error codes 209
example 209
execution conditions 208
fault conditions 207
operands 206
Motion Group Stop (MGS) 31, 196-201
arithmetic status flags 199
changes to status bits 201
description 197
error codes 201
example 201
execution conditions 200
fault conditions 199
operands 196

Motion Group Strobe Position (MGSP) 31, 210-213
arithmetic status flags 211
changes to status bits 213
description 211
error codes 213
example 213
execution conditions 212
fault conditions 211
operands 210
Motion Move Instructions
Motion Axis Gear (MAG) 31, 123-132
Motion Axis Home (MAH) 31, 92-98
Motion Axis Jog (MAJ) 31, 99-110
Motion Axis Move (MAM) 31, 111-121
Motion Axis Position Cam (MAPC) 31, 156-177
Motion Axis Stop (MAS) 31, 82-88
Motion Axis Time Cam (MATC) 31, 178-191
Motion Calculate Cam Profile (MCCP) 31, 150-155
Motion Calculate Slave Values (MCSV) 31, 192-194
Motion Change Dynamics (MCD) 31, 133-141
Motion Redefine Position (MRP) 31, 142-149
Motion Redefine Position (MRP) 31, 142-149
arithmetic status flags 146
changes to status bits 148
description 143
error codes 148
execution conditions 147
fault conditions 146
operands 142
Motion Run Axis Tuning (MRAT) 32, 268-277
arithmetic status flags 275
changes to status bits 277
description for use with AXIS_CIP_DRIVE 272, 289
description for use with AXIS_SERVO or AXIS_SERVO_DRIVE 269
error codes 277
execution conditions 276
fault conditions 275
operands 268
tune status parameter 274
Motion Run Hookup Diagnostics (MRHD) 32, 284-296
arithmetic status flags 294
changes to status bits 296
Commutation test 293
description for use with AXIS_SERVO or AXIS_SERVO_DRIVE 286
Encoder Hookup test 287, 291
error codes 296
execution conditions 295
fault conditions 294
Marker Hookup test 288, 292
Motor Encoder Hookup test 287, 290
operands 284
test status 288, 293
Motion Servo Off (MSF) 31, 43-47
arithmetic status flags 45
changes to status bits 47
description 44
error codes 46
example 47
execution conditions 46
fault conditions 45
operands 43
Motion Servo On (MSO) 31, 39-42
arithmetic status flags 40
changes to status bits 42
description 40
error codes 41
execution conditions 41
extended error codes 42
fault conditions 40
operands 39
Motion State Instructions 35-79
Motion Axis Fault Reset (MAFR) 31, 75-79
Motion Axis Shutdown (MASD) 31, 48-53
Motion Axis Shutdown Reset (MASR) 31, 54-58
Motion Direct Drive Off (MDF) 31, 65-68
Motion Direct Drive On (MDO) 31, 59-64
Motion Drive Start (MDS) 31, 69-74
Motion Servo Off (MSF) 31
Motion Servo On (MSO) 31, 39-42
Motion State Instructions
Motion Servo Off (MSF) 47
Motion State Instructions
Motion Servo Off (MSF) 43
MOTION_GROUP data type 354
MOTION_INSTRUCTION data type 356
MRP 405
Multi-Axis Coordinated Motion Instructions 391
Introduction 392
multiple watch position events 218

N

no master axis 305
non-linear motion 395

O

operational cam 167, 185
OUTPUT_CAM data type 357
OUTPUT_COMPENSATION data type 358

P

path fidelity 325
pending 403, 405, 407, 408, 411
position cam 411
pending cam 167, 185, 409
pending cams 400
point pair 394
position cam 401, 404, 409, 411
position cam profile 395

position-lock cams 395
process motion instructions 29
profile 397

R

RampAcceleration 69
RampDeceleration 69
RampJerk - Control 69
RampVelocity - Negative 69
RampVelocity - Positive 69
reciprocating motion 393
reverse only 403
run 397
run cam 398

S

salve axis 411
scaling 401
S-curve profile
troubleshoot 331-343
tune 327-329
slave
actual 304
command 304
relationship 302
slave axis 393, 397, 404
acceleration cam profile 397
execution mode 403
move process 396
position values 395
time cam profile 396
slave axis position 409
slave1 298
speed unit 314
speed units 318
status bit 404
stopping
on the slave 307
structured text programming 359-387
assignments 360-362
CASE...OF 375-377
comments 387
contructs 370
expressions 362-368
FOR…DO 378-380
IF...THEN 371-374
REPEAT…UNTIL 384-386
WHILE…DO 381-383
structures
See data type

T

time based planning 323
time cam profile 395, 396
time cam status 408
troubleshoot
axis accelerates to a stop 331
axis overshoots its position and reverses direction 342
axis overshoots the target speed 334
axis reverses direction when stopped and started 338
delay if a jog stopped and restarted 336
jerk 331-343
S-curve profile 331-343
using error codes 345-352
tune
jerk 327-329
S-curve profile 327-329
types of camming 393

Notes:

Notes:

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At http://www.rockwellautomation.com/support you can find technical and application notes, sample code, and links to software service packs. You can also visit our Support Center at https://rockwellautomation.custhelp.com/ for software updates, support chats and forums, technical information, FAQs, and to sign up for product notification updates.

In addition, we offer multiple support programs for installation, configuration, and troubleshooting. For more information, contact your local distributor or Rockwell Automation representative, or visit http://www.rockwellautomation.com/services/online-phone.

Installation Assistance

If you experience a problem within the first 24 hours of installation, review the information that is contained in this manual. You can contact Customer Support for initial help in getting your product up and running.

United States or Canada	1.440.646.3434
Outside United States or Canada	Use the Worldwide Locator at http://www.rockwellautomation.com/rockwellautomation/support/overview.page, or contact your local Rockwell Automation representative.

New Product Satisfaction Return

Rockwell Automation tests all of its products to help ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

United States	Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for the return procedure.

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete this form, publication RA-DU002, available at http://www.rockwellautomation.com/literature/.

Rockwell Automation maintains current product environmental information on its website at http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page.

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication MOTION-RM002E-EN-P - July 2015

Supersedes Publication MOTION-RM002D-EN-P - September 2014

Copyright © 2015 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.